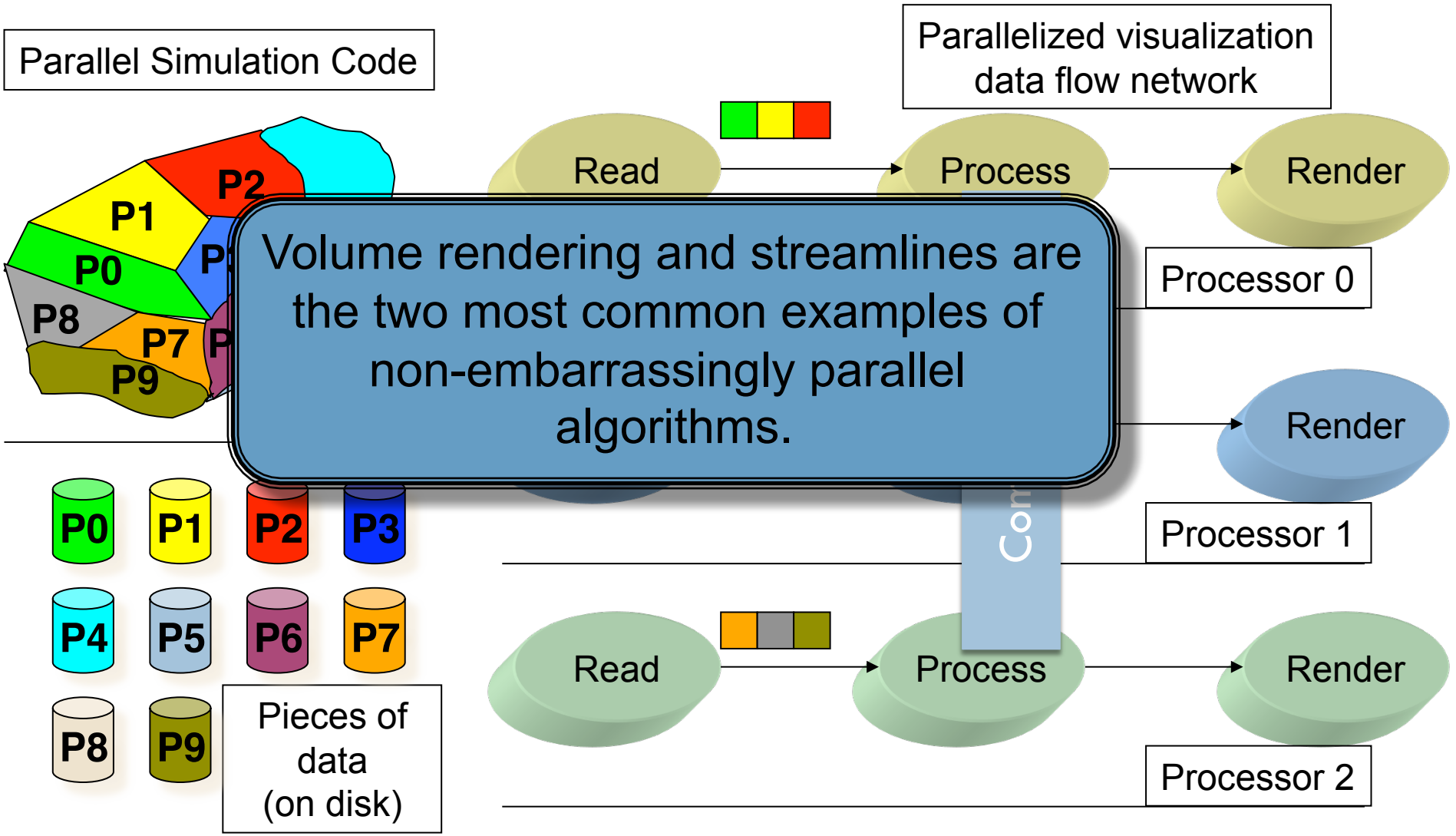
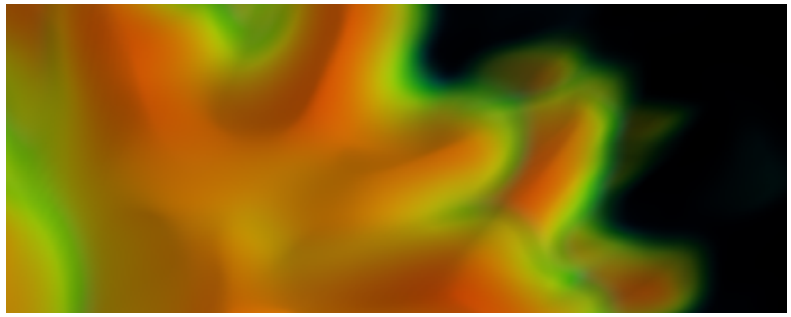
How do we handle algorithms that aren't embarrassingly parallel?





MPI-hybrid Parallelism for Volume Rendering on Large, Multi-core Systems

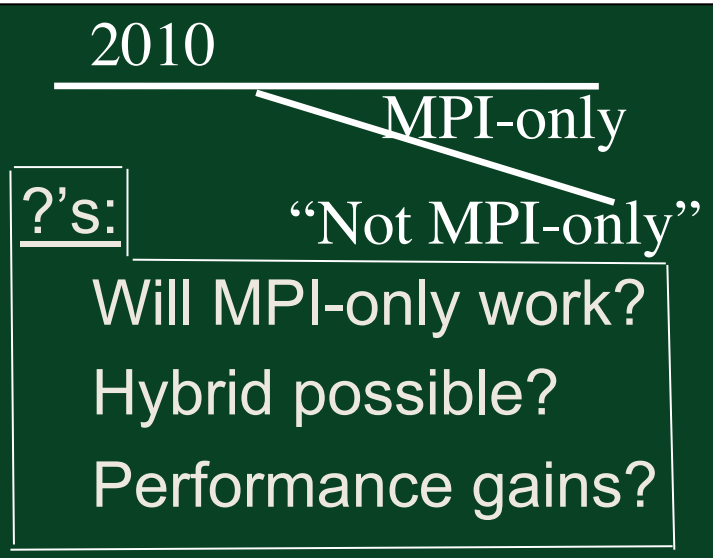
Mark Howison, E. Wes Bethel, and Hank Childs
Lawrence Berkeley National Laboratory

EGPGV 2010
Norrköping, Sweden

Hybrid Parallelism for Volume Rendering on Large, Multi-core Platforms

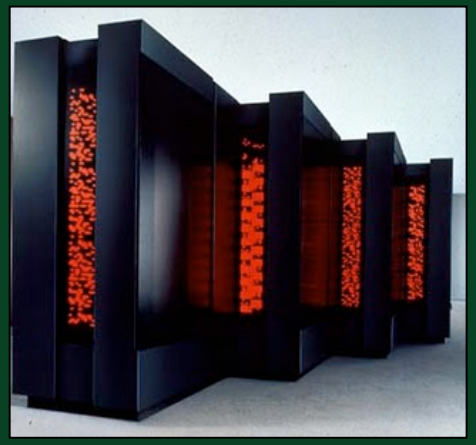
Overview

- Traditional approaches for parallel visualization may not work well in future: 100-1000 cores per node.
  - Exascale machines will likely have O(1M) nodes
- Hybrid-parallelism blends distributed- and shared-memory parallelism concepts.
- This study:
  - Does hybrid-parallelism work for volume rendering at extreme concurrency? If so, how well?
  - Experiment to compare performance shows favorable characteristics of hybrid-parallel, especially at very high concurrency.



Parallelism

- Mid 1970s-Early 1990s:
  - Vector machines: Cray 1 ... NEC SX
  - Vectorizing Fortran compilers help optimize a[i]=b[i]*x+c.
- Early 1990s-present:
  - The rise of the MPP based on the commodity microprocessor. Cray T3D, TM CM1, CM2, CM5, etc.
  - Message Passing Interface (MPI) becomes the gold standard for building/running parallel codes on MPPs.
- Early 1990s-Early 2000s:
  - Shared memory parallelism (e.g. SGI)
- Mid 2000s-present:
  - Rise of the multi-core CPU, GPU. AMD Opteron, Intel Nehalem, Sony Cell BE, NVIDIA G80, etc.
  - Large supercomputers comprised of lots of multi-core CPUs.
  - Shared memory programming on a node: pthreads, OpenMP; data parallel languages (CUDA); global shared memory languages (UPC) and utilities (CAF).



Related work in Hybrid Parallelism

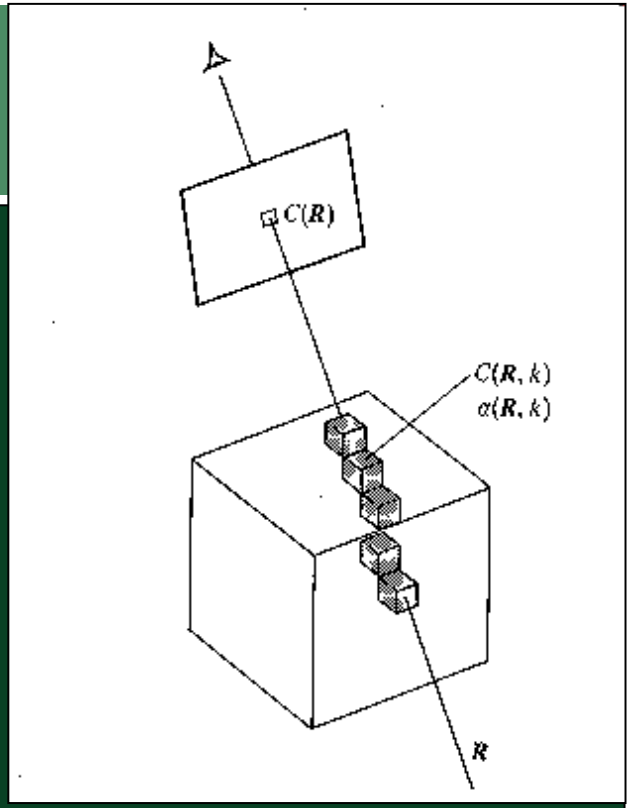
- Caveats
  - Relatively new research area, not a great deal of published work.
  - Studies focus on “solvers,” not vis/graphics.
  - State of hybrid parallel visualization: lots of work to do
- Fundamental questions:
  - How to map algorithm onto a complex memory, communication hierarchy?
  - What is the right balance of distributed- vs. shared-memory parallelism? How does balance impact performance?
- Conclusions of these previous works:
  - What is best? Answer: it depends.
  - Many factors influence performance/scalability:
    - Synchronization overhead.
    - Load balance (intra- and inter-node).
    - Communication overhead and patterns.
    - Memory access patterns.
    - Fixed costs of initialization.
    - Number of runtime threads.

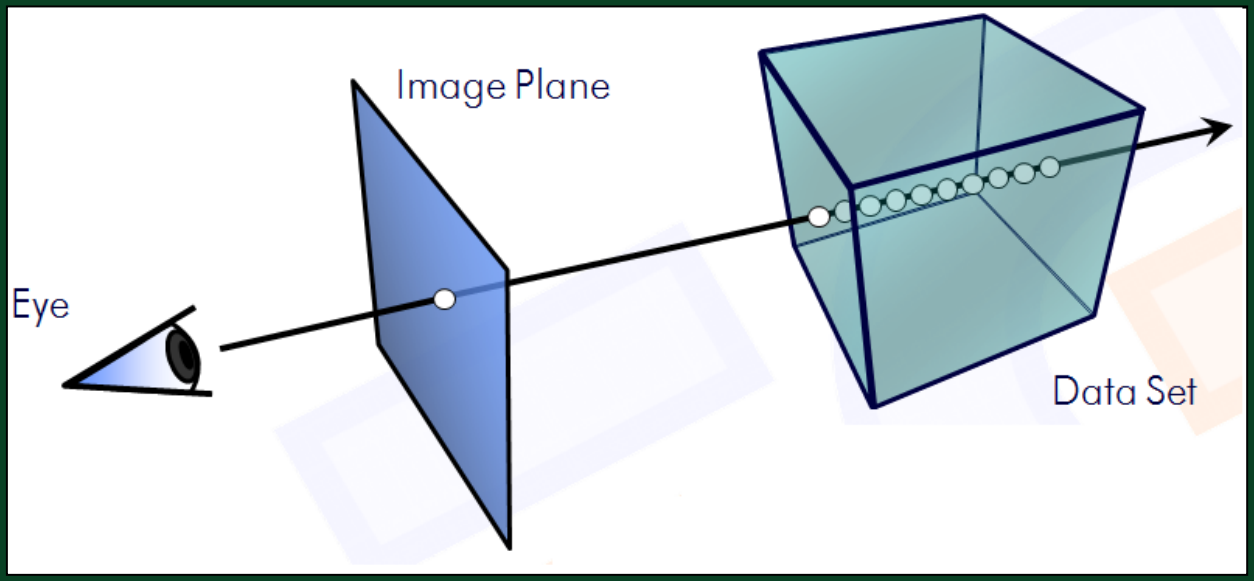
This Study

- Hybrid parallelism on visualization: raycasting volume rendering.
  - Ask same questions the HPC folks do:
    - How to map algorithm to hybrid parallel space?
    - How does performance compare with MPI-only implementation?
- Hybrid-parallel implementation/architecture.
- Performance study.
  - Runs at 216K-way parallel
  - Look at:
    - Costs of initialization.
    - Memory use comparison.
    - Scalability.
    - Absolute runtime.

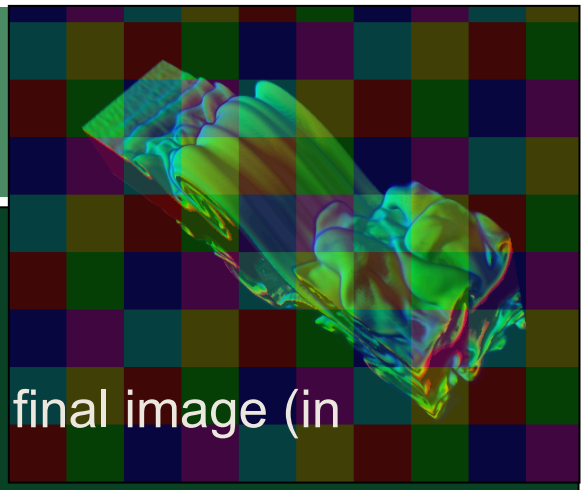
Algorithm Studied: Raycasting VR

- Overview of Levoy's method
  - For each pixel in image plane:
    - Find intersection of ray and volume
    - Sample data (RGBa) along ray, integrate samples to compute final image pixel color

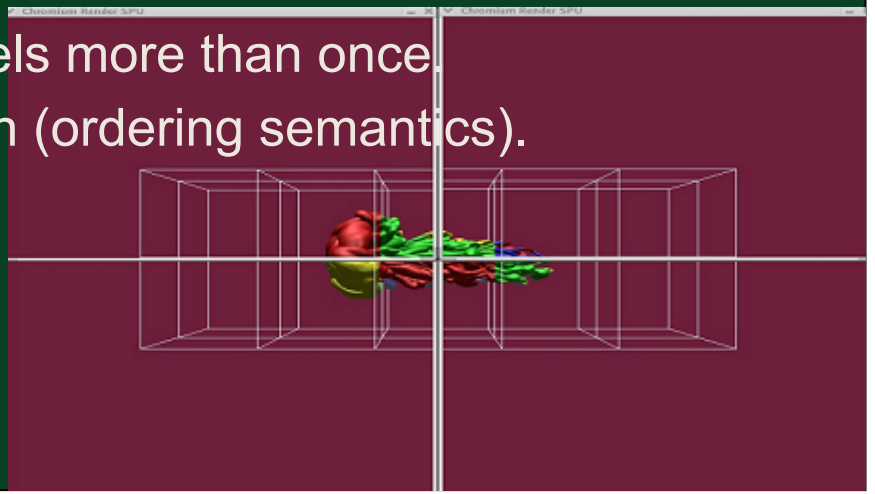




Parallelizing Volume Rendering



- Image-space decomposition.
  - Each process works on a disjoint subset of the final image (in parallel)
  - Processes may access source voxels more than once, will access a given output pixel only once.
  - Great for shared memory parallelism.
- Object-space decomposition.
  - Each process works on a disjoint subset of the input data (in parallel).
  - Processes may access output pixels more than once.
  - Output requires image composition (ordering semantics).



Hybrid Volume Rendering

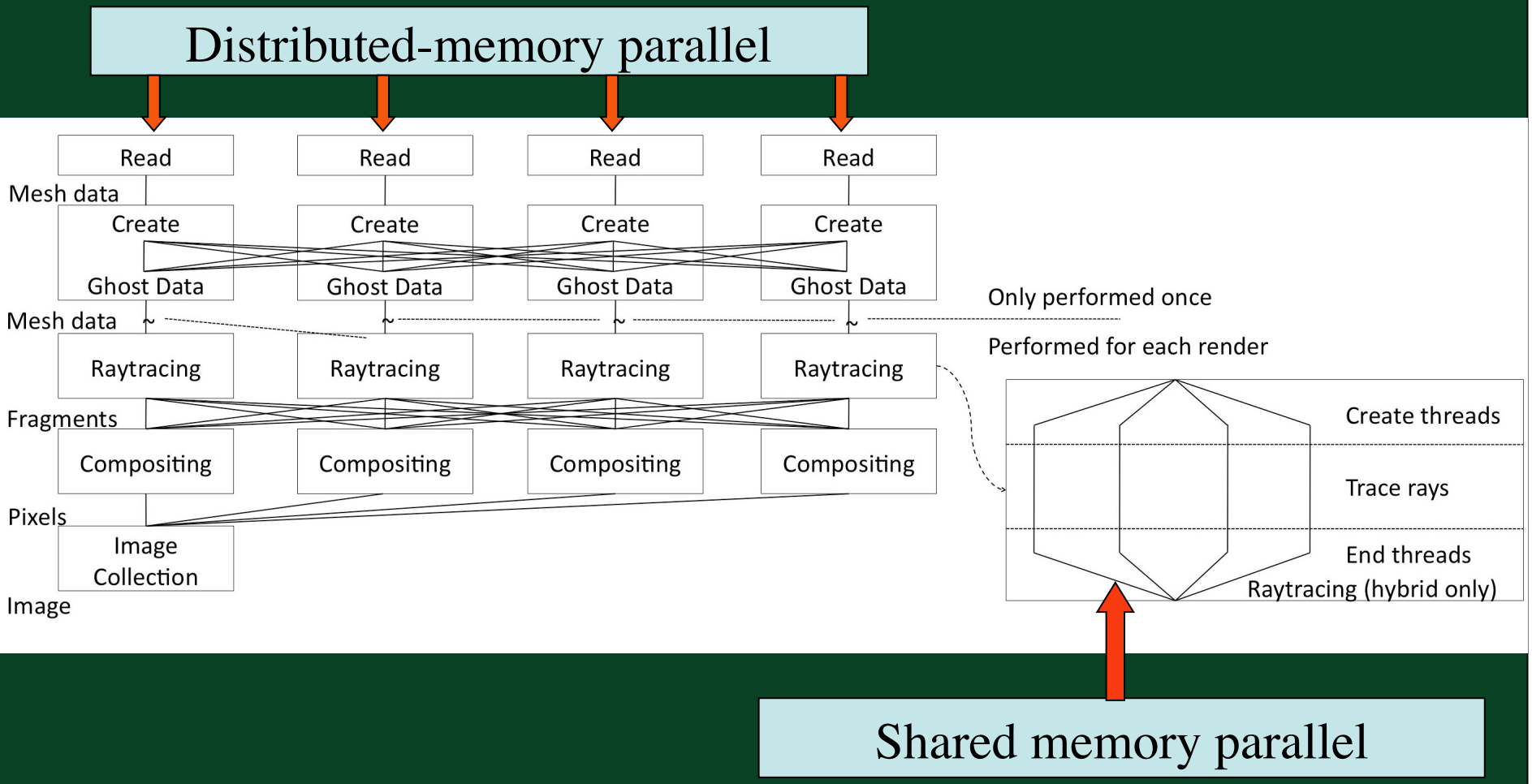
- Hybrid volume rendering:
  - Refers to mixture of object- and image-order techniques to do volume rendering.
  - Most contemporary parallel volume rendering projects are hybrid volume renderers:
    - Object order – divide data into disjoint chunks, each processor works on its chunk of data.
    - Image order – parallel compositing algorithm divides work over final image, each composites over its portion of the final image.
    - A two-stage algorithm, heavy communication load between stages.

Hybrid Parallel Volume Rendering

- Hybrid-parallelism a blend of shared- and distributed-memory parallelism.
- Details of hybrid parallel implementation described on the next slide
  - 2 Implementations: pthreads, OpenMP.
- Note the difference between hybrid parallel volume rendering and hybrid volume rendering

Hybrid Parallel Volume Rendering

- Our hybrid-parallel architecture:

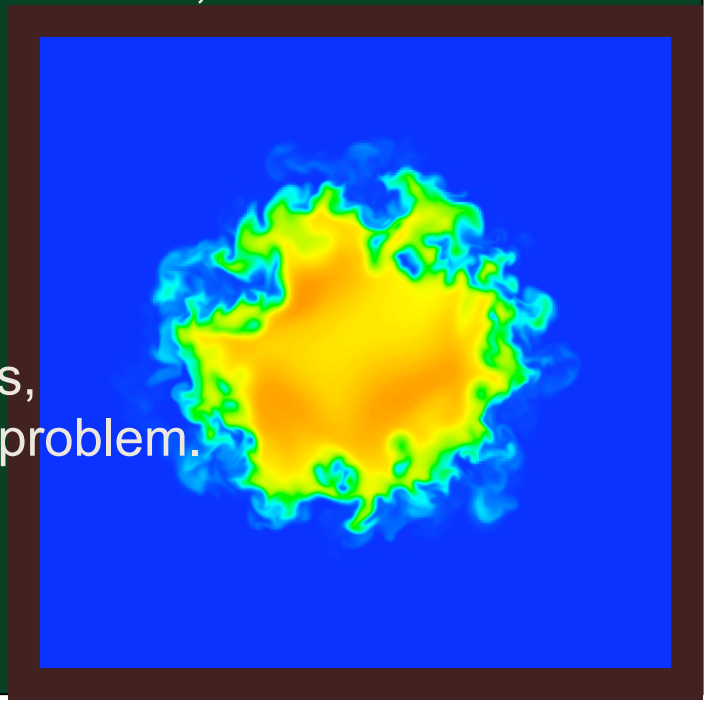


Our Experiment

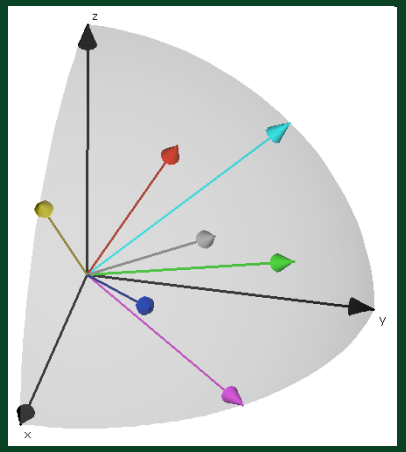
- Thesis: hybrid-parallel will exhibit favorable performance, resource utilization characteristics compared to traditional approach.
- How/what to measure?
  - Memory footprint, scalability characteristics, absolute runtime.
  - Across a wide range of concurrencies.
    - Remember: we're concerned about what happens at extreme concurrency.
  - Also varied view point to induce different memory access patterns.
- Strong scaling study: hold problem size constant, vary amount of resources.

Experiment: Platform and Source Data

- Platform: JaguarPF, a Cray XT5 system at ORNL
  - 18,688 nodes, dual-socket, six-core AMD Opteron (224K cores)
- Source data:
  - Combustion simulation results, hydrogen flame (data courtesy J. Bell, CCSE, LBNL)
  - Effective AMR resolution: 1024^3, flattened to 512^3, runtime upscaled to 4608^3 (to avoid I/O costs).
- Target image size: 4608^2 image.
  - Want approx 1:1 voxels to pixels.
- Strong scaling study:
  - As we increase the number of procs/cores, each proc/core works on a smaller-sized problem.
  - Time-to-solution should drop.



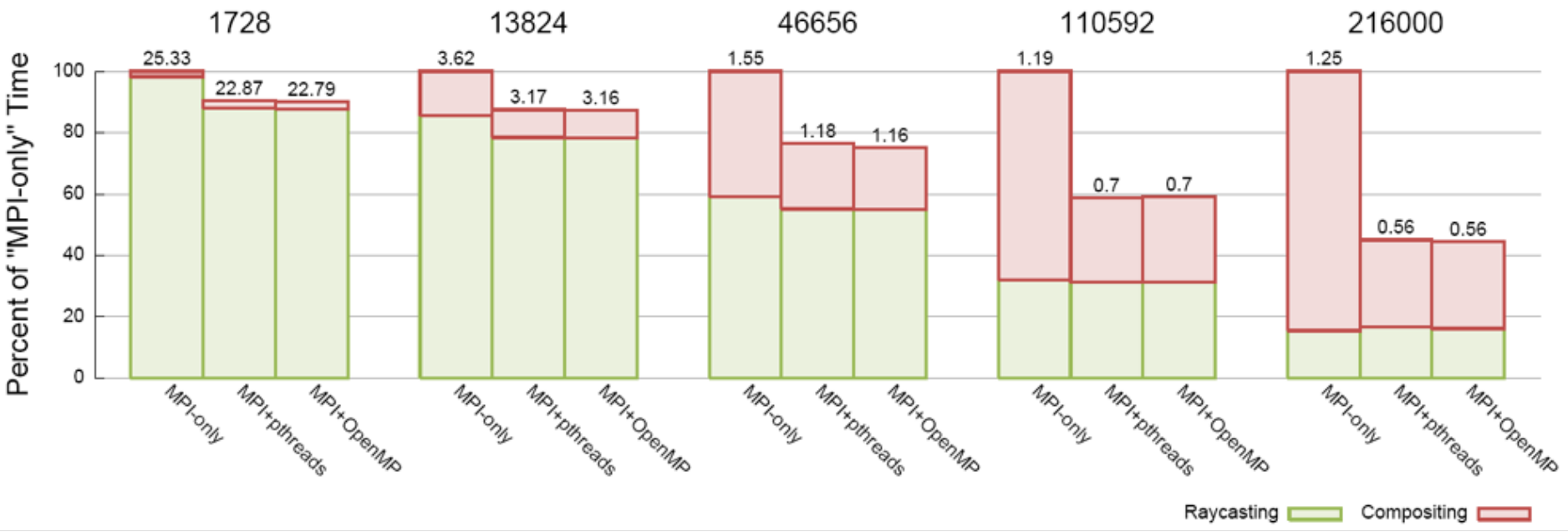
Experiment – The Unit Test



- Raycasting time: view/data dependent
  - Execute from 10 different prescribed views: forces with- and cross-grained memory access patterns.
  - Execute 10 times, result is average of all.
- Compositing
  - Five different ratios of compositing PEs to rendering PEs.
- Measure:
  - Memory footprint right after initialization.
  - Memory footprint for data blocks and halo exchange.
  - Absolute runtime and scalability of raycasting and compositing.

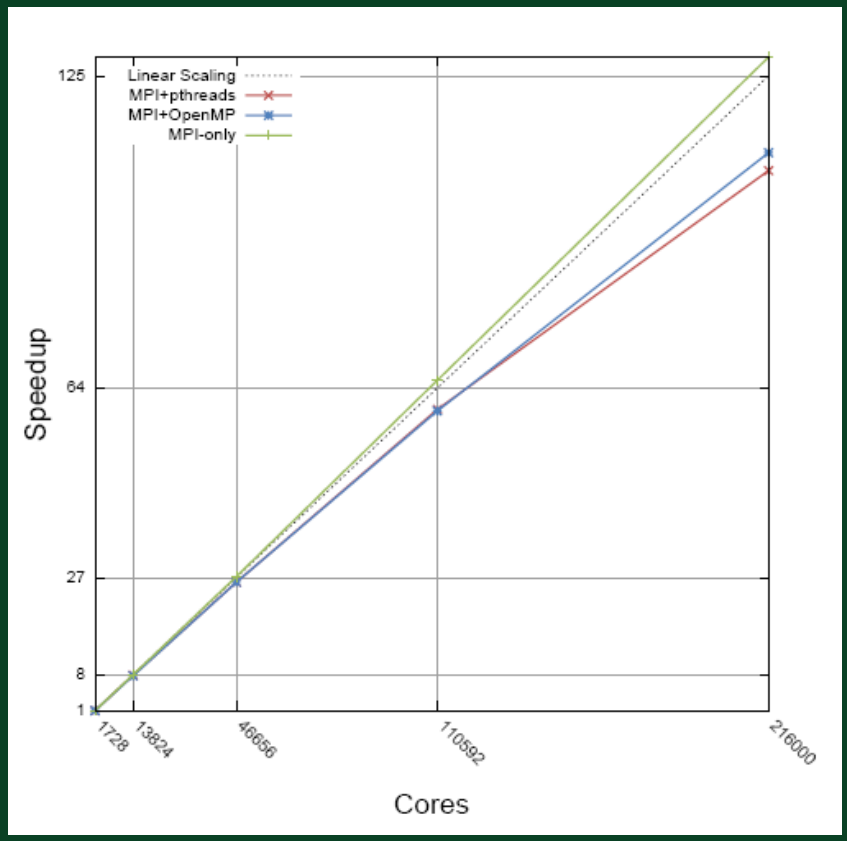
Absolute Runtime

- -hybrid outperforms –only at every concurrency level.
  - At 216K-way parallel, -hybrid is more than twice as fast as –only.
  - Compositing times begin to dominate: communication costs.



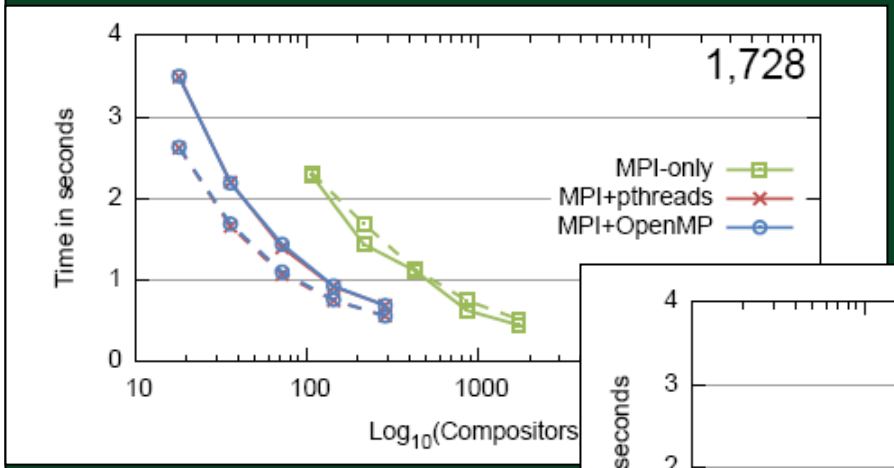
Scalability – Raycasting Phase

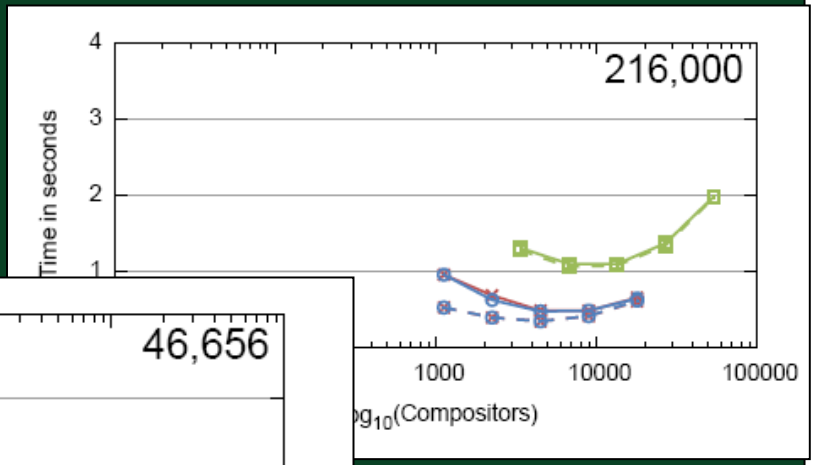
- Near linear scaling since no interprocess communication.
- -hybrid shows sublinear scaling due to oblong block shape.
- -only shows slightly better than linear due to reduced work caused by perspective foreshortening.

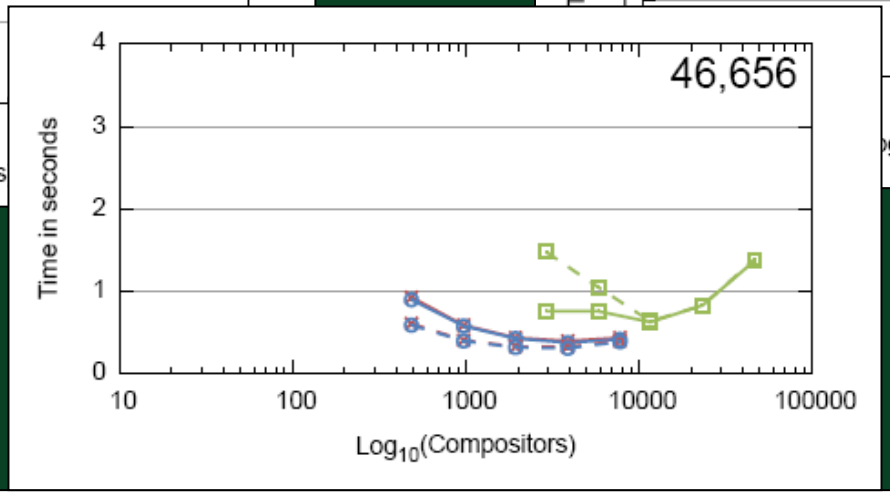


Scalability – Compositing

- How many compositors to use?
  - Previous work: 1K to 2K for 32K renderers (Peterka, 2009).
  - Our work: above ~46K renderers, 4K to 8K works better.
  - -hybrid cases always performs better: fewer messages.
  - Open question: why the critical point?







Memory Use – Data Decomposition

- 16GB RAM per node
  - Sets lower bound on concurrency for this problem size: 1728-way parallel (no virtual memory!).
- Source data (1x), gradient field (3x)
- Want cubic decomposition.
  - 1x2x3 block configuration per socket for –only.
- -hybrid has ~6x data per socket than –only
  - Would prefer to run study on 8-core CPUs to maintain cubic shape

MPI-only		MPI-hybrid		
MPI PEs	Block Dimensions	MPI PEs	Block Dimensions	Memory Per Node
12^3=1728	$384 \times 384 \times 384$	288	$384 \times 768 \times 1152$	10368MB
24^3=13824	$192 \times 192 \times 192$	2304	$192 \times 384 \times 576$	1296MB
36^3=46656	$128 \times 128 \times 128$	7776	$128 \times 256 \times 384$	384MB
48^3=110592	$96 \times 96 \times 96$	18432	$96 \times 192 \times 288$	162MB
60^3=216000	$76 \times 76 \times 76$	36000	$76 \times 153 \times 230$	80.4MB / 81.6MB

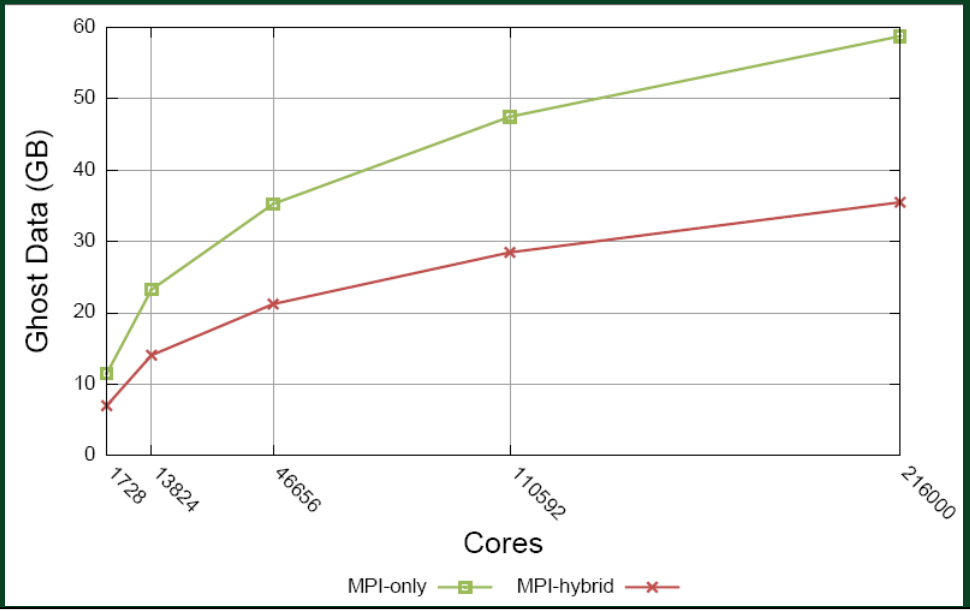
Memory Use – MPI_Init()

- Per PE memory:
  - About the same at 1728, over 2x at 216000.
- Aggregate memory use:
  - About 6x at 1728, about 12x at 216000.
  - At 216000, -only requires 2GB of memory for initialization per node!!!

Cores	Mode	MPI PEs	MPI Runtime Memory Usage		
			Per PE (MB)	Per Node (MB)	Aggregate (GB)
1728	MPI-hybrid	288	67	133	19
1728	MPI-only	1728	67	807	113
13824	MPI-hybrid	2304	67	134	151
13824	MPI-only	13824	71	857	965
46656	MPI-hybrid	7776	68	136	518
46656	MPI-only	46656	88	1055	4007
110592	MPI-hybrid	18432	73	146	1318
110592	MPI-only	110592	121	1453	13078
216000	MPI-hybrid	36000	82	165	2892
216000	MPI-only	216000	176	2106	37023

Memory Use – Ghost Data

- Two layers of ghost cells required for this problem:
  - One for trilinear interpolation during ray integration loop.
  - Another for computing a gradient field (central differences) for shading.
- Hybrid approach uses fewer, but larger data blocks.
  - ~40% less memory required for ghost data (smaller surface area)
  - Reduced communication costs



Comparing our results to classic hybrid parallel factors

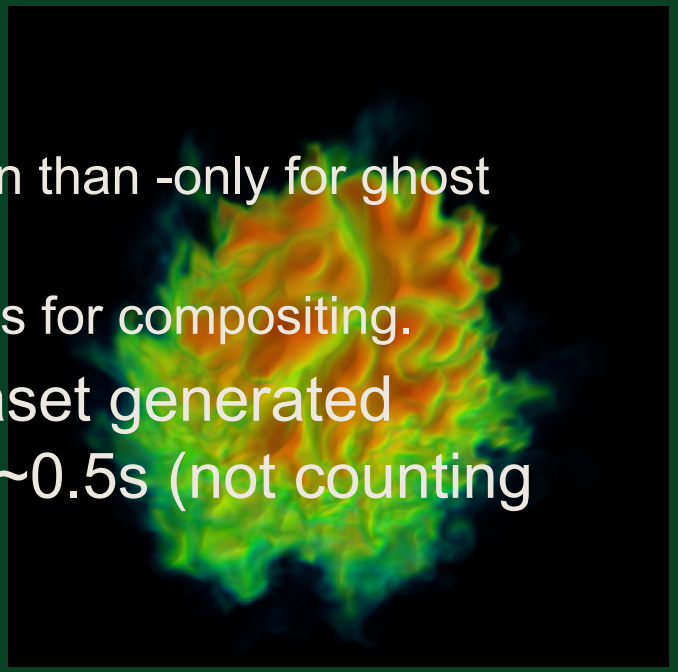
- Factors in hybrid parallelism performance
  - Synchronization overhead.
    - Had two MPI tasks per node, not one, to prevent work spreading across CPU.
  - Load balance (intra- and inter-node).
    - Studied extensively, comes down to communication
  - Communication overhead and patterns.
    - Hybrid implementation naturally lends itself to superior communication pattern
  - Memory access patterns.
    - Not presented
  - Fixed costs of initialization.
    - Ghost data generation cost reduced with hybrid parallelism
    - MPI initialization cost reduced with hybrid parallelism
  - Number of runtime threads.
    - Not studied

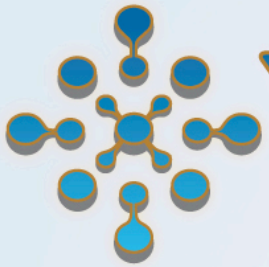
3 Questions revisited

- Seeing poor indicators @ 216K
- Yes
- Yes

Summary of Results

- Absolute runtime: -hybrid twice as fast as –only at 216K-way parallel.
- Memory footprint: -only requires 12x more memory for MPI initialization then –hybrid
  - Factor of 6x due to 6x more MPI PEs.
  - Additional factor of 2x at high concurrency, likely a vendor MPI implementation (an N^2 effect).
- Communication traffic:
  - -hybrid performs 40% less communication than -only for ghost data setup.
  - -only requires 6x the number of messages for compositing.
- Image: 4608^2 image of a ~4500^3 dataset generated using 216,000 cores on JaguarPF in ~0.5s (not counting I/O time).





VisWeek 2010

VIS • INFOVIS • VAST

Large Vector-Field Visualization, Theory and Practice:

Large Data and Parallel Visualization

Hank Childs +

D. Pugmire, D. Camp, C. Garth,
G. Weber, S. Ahern, & K. Joy

Lawrence Berkeley National Laboratory /

University of California at Davis

October 25, 2010

BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

UCDAVIS
UNIVERSITY OF CALIFORNIA

IDAV Institute for Data Analysis and Visualization

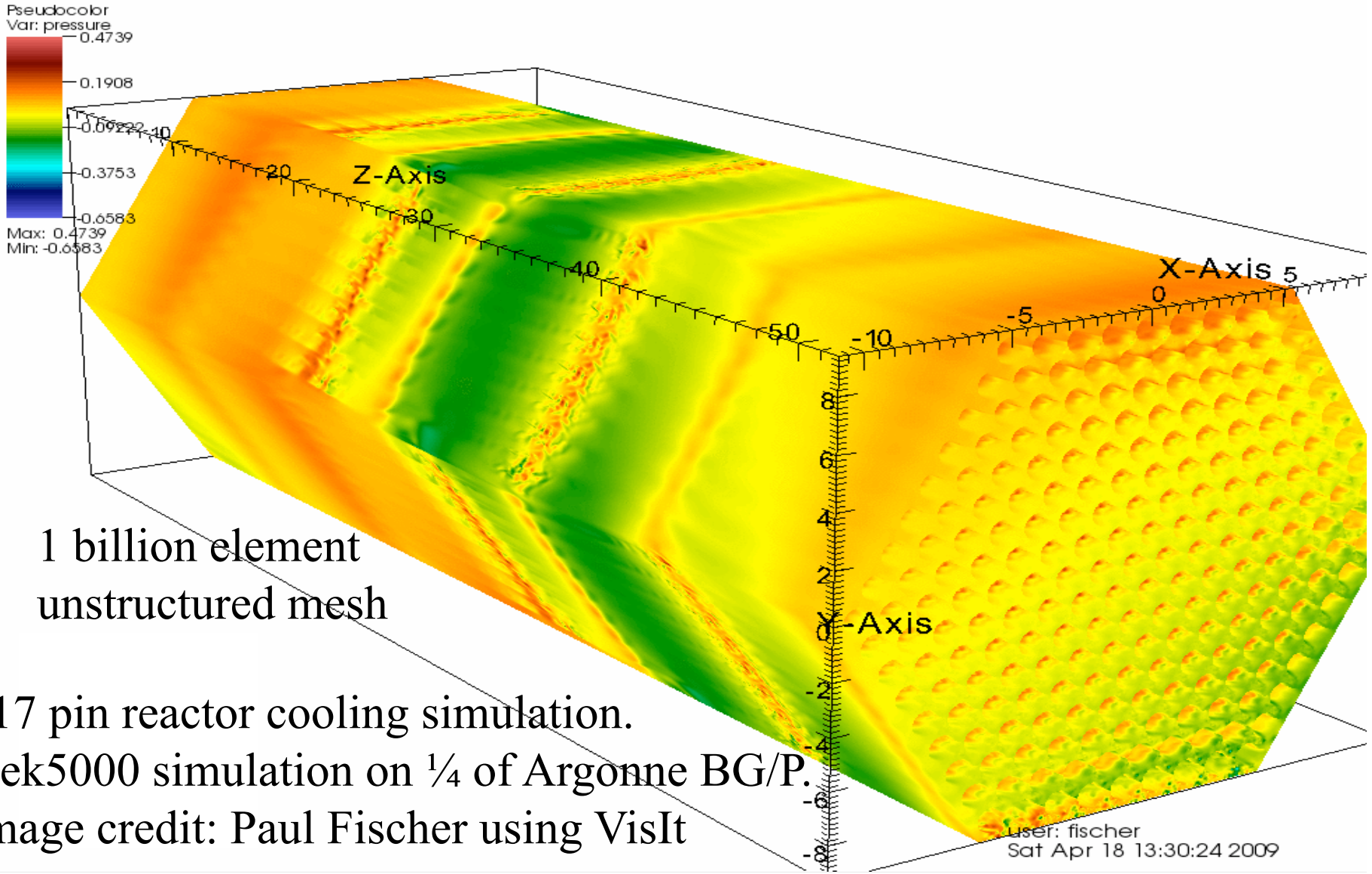
Outline

- Motivation
- Parallelization strategies
- Master-slave parallelization
- Hybrid parallelism

Outline

- Motivation
- Parallelization strategies
- Master-slave parallelization
- Hybrid parallelism

Supercomputers are generating large data sets that often require parallelized postprocessing.



1 billion element
unstructured mesh

217 pin reactor cooling simulation.
Nek5000 simulation on ¼ of Argonne BG/P.
Image credit: Paul Fischer using VisIt

Communication between “channels” are a key factor in effective cooling.

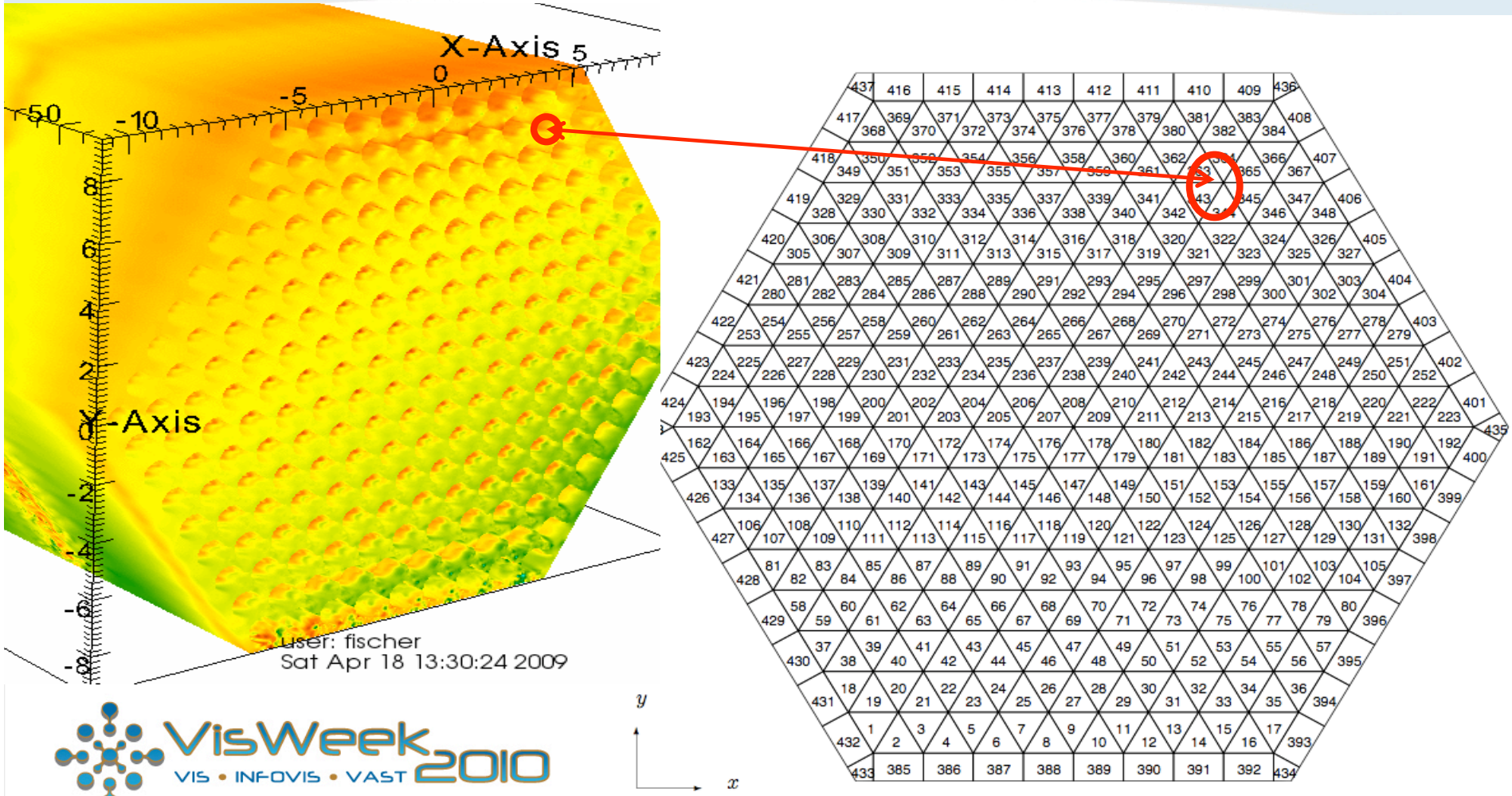
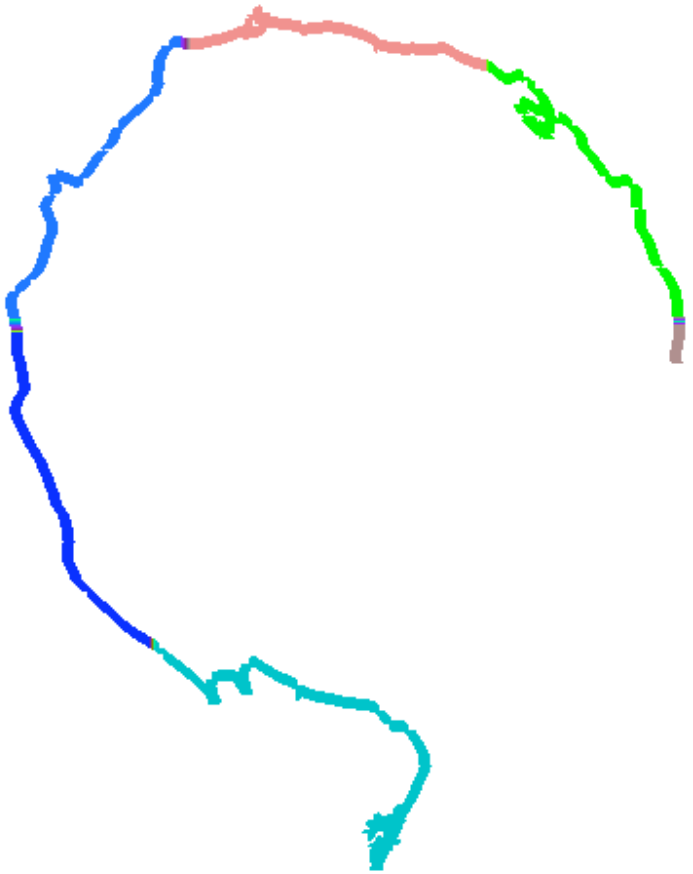
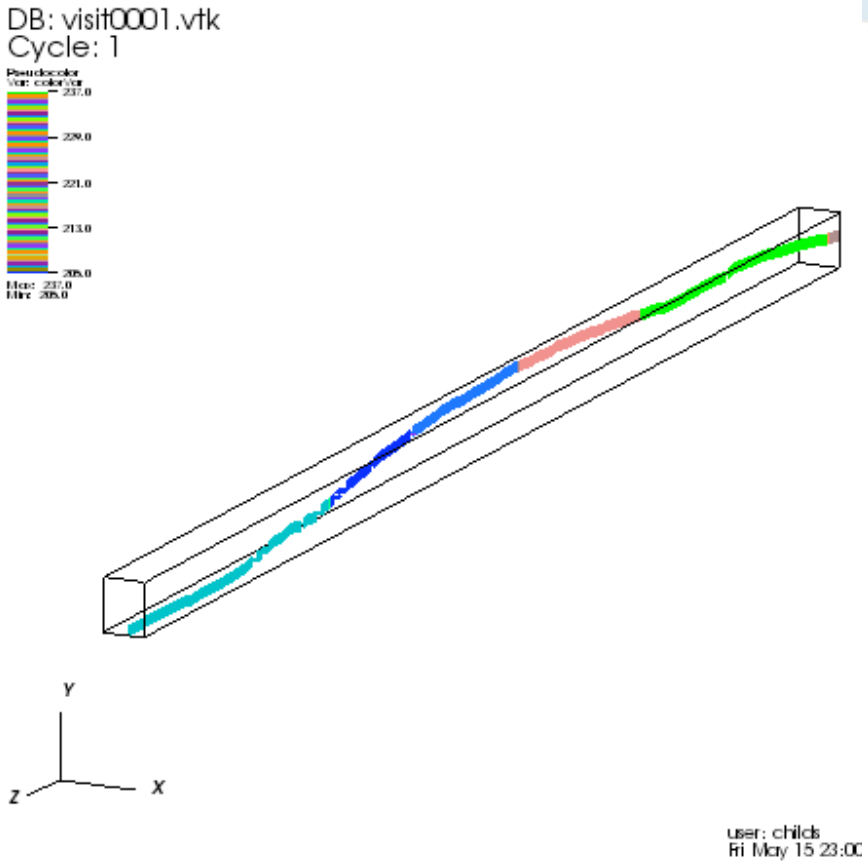


Figure 1: cells_only.ps

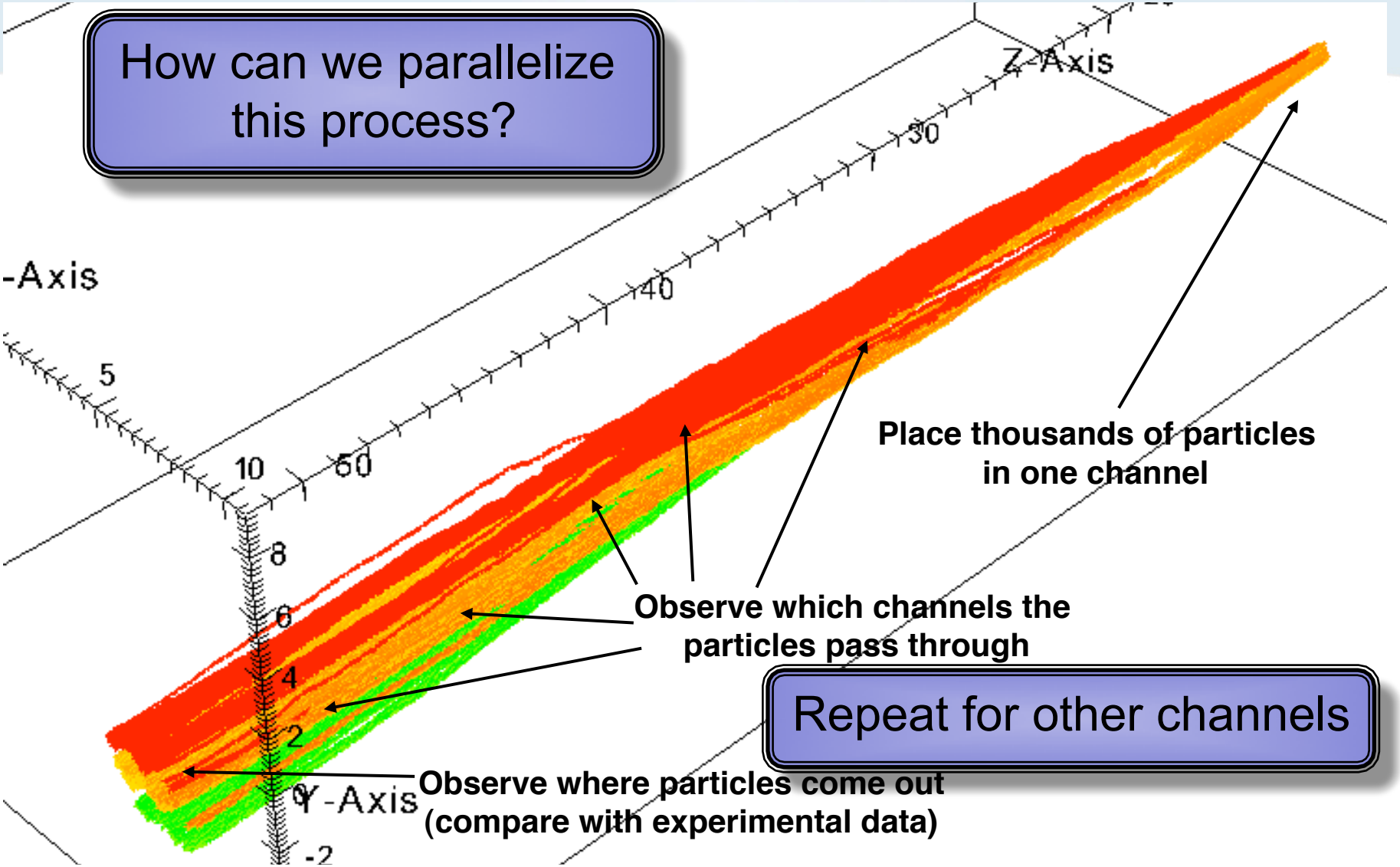
Particle advection can be used to study communication properties.





VisWeek 2010
VIS • INFOVIS • VAST

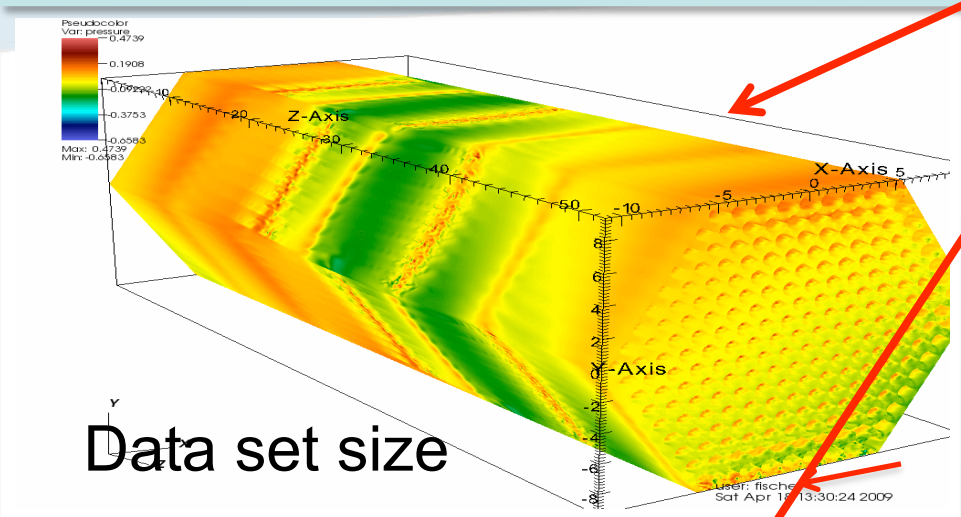
This sort of analysis requires many particles to be statistically significant.



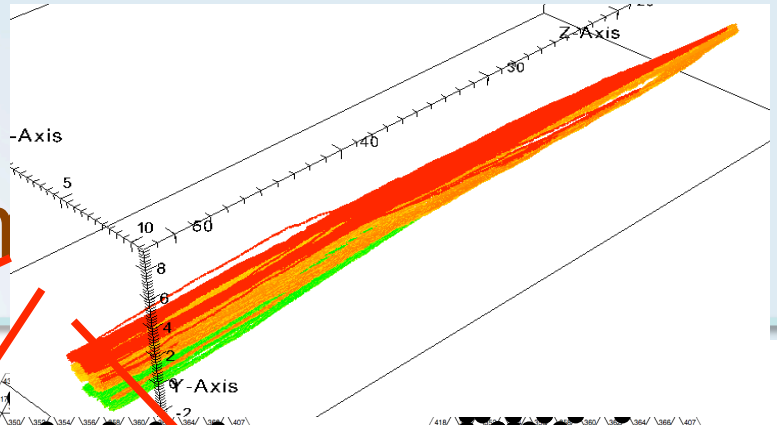
Outline

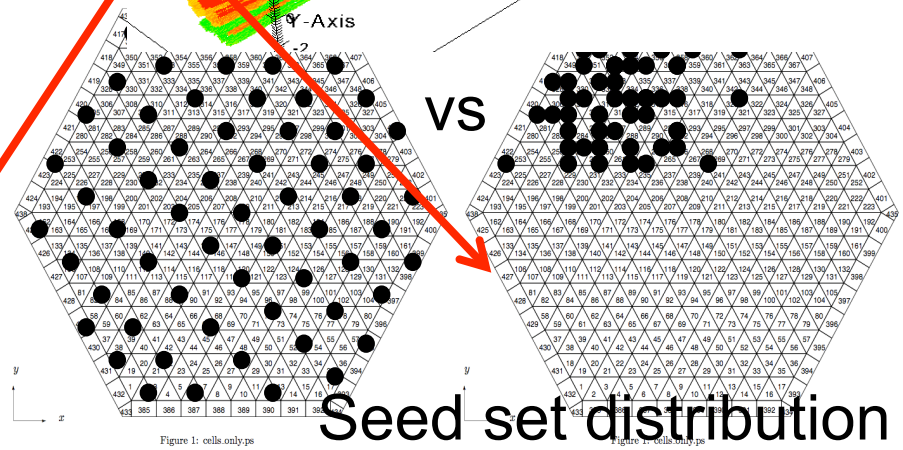
- Motivation
- Parallelization strategies
- Master-slave parallelization
- Hybrid parallelism

Particle advection: Four dimensions of con

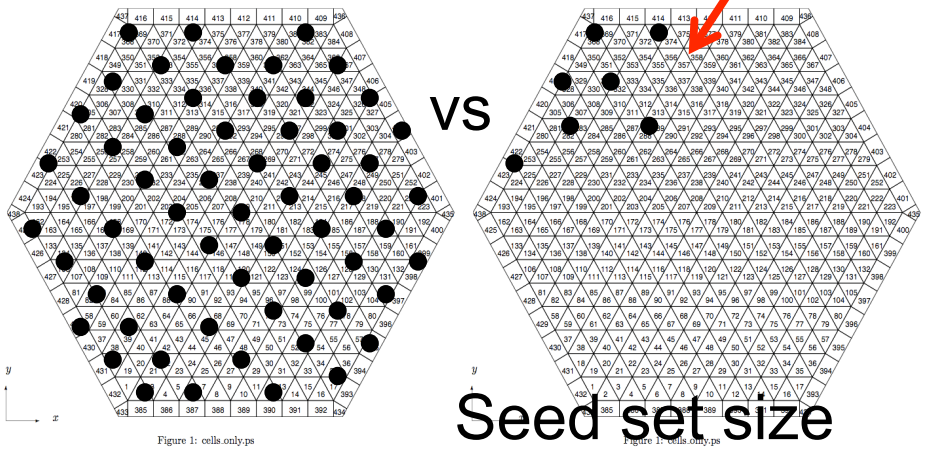


Data set size

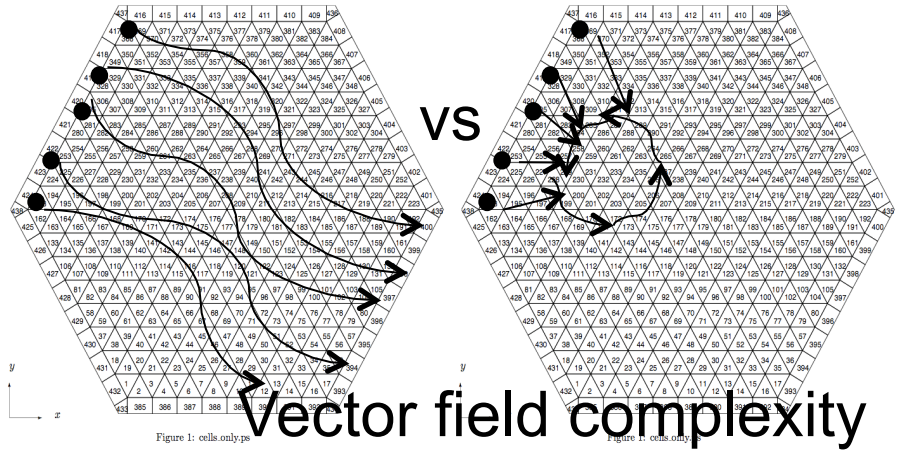




Seed set distribution



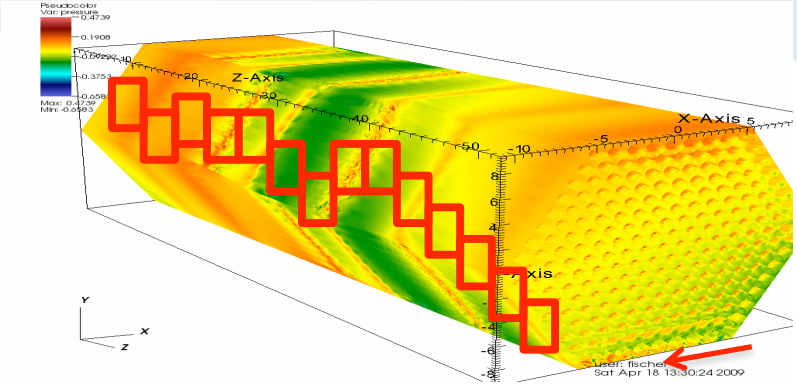
Seed set size



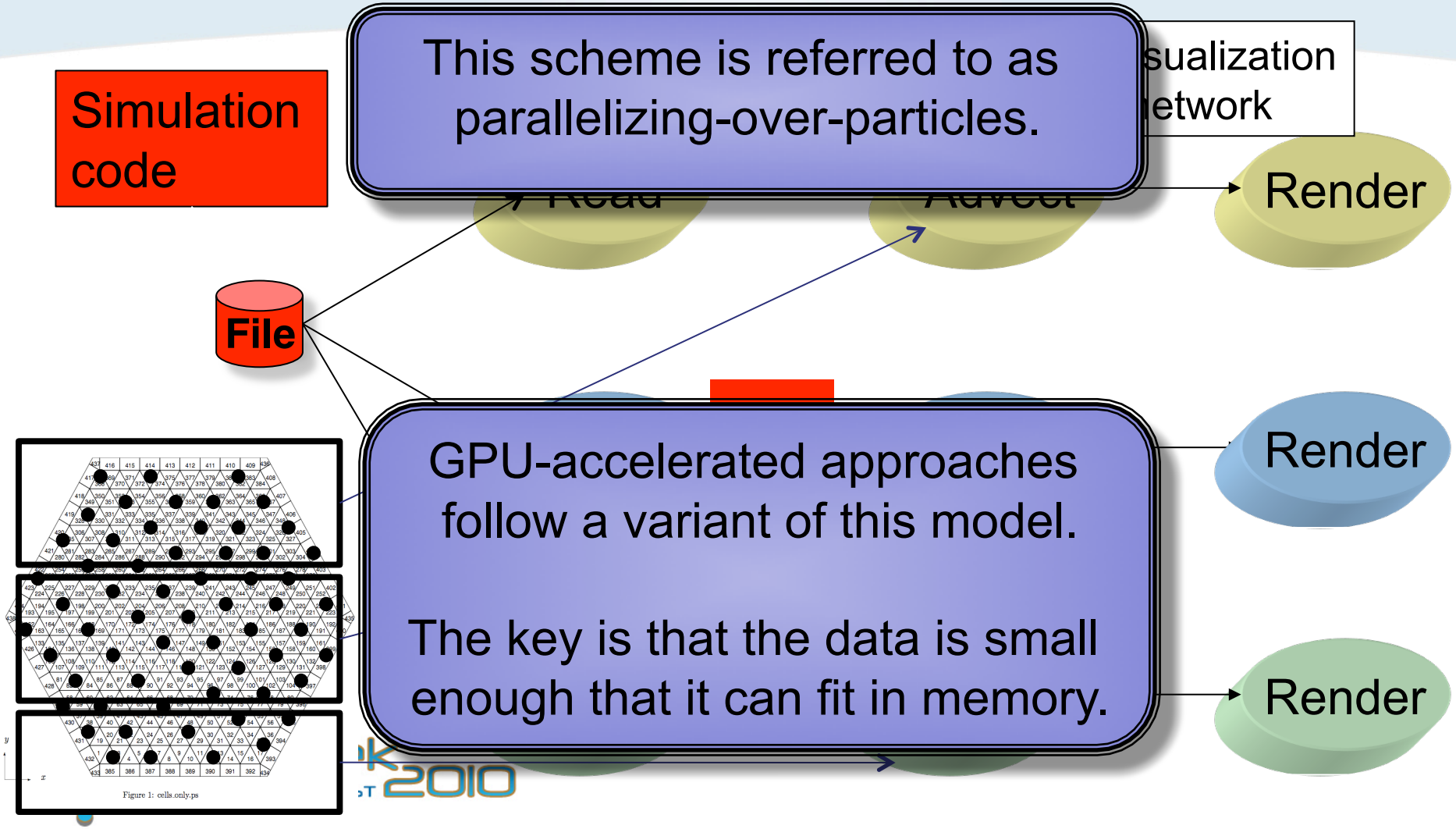
Vector field complexity

Do we need parallel processing? When? How complex?

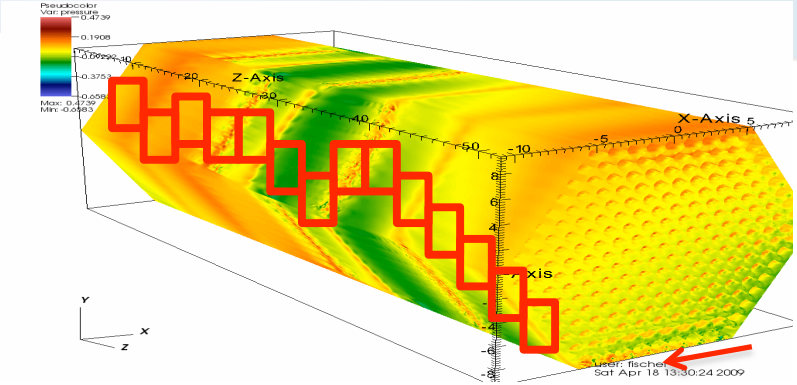
- Data set size?
  - Not enough!
- Large #'s of particles?



Parallelization for small data and a large number of particles.

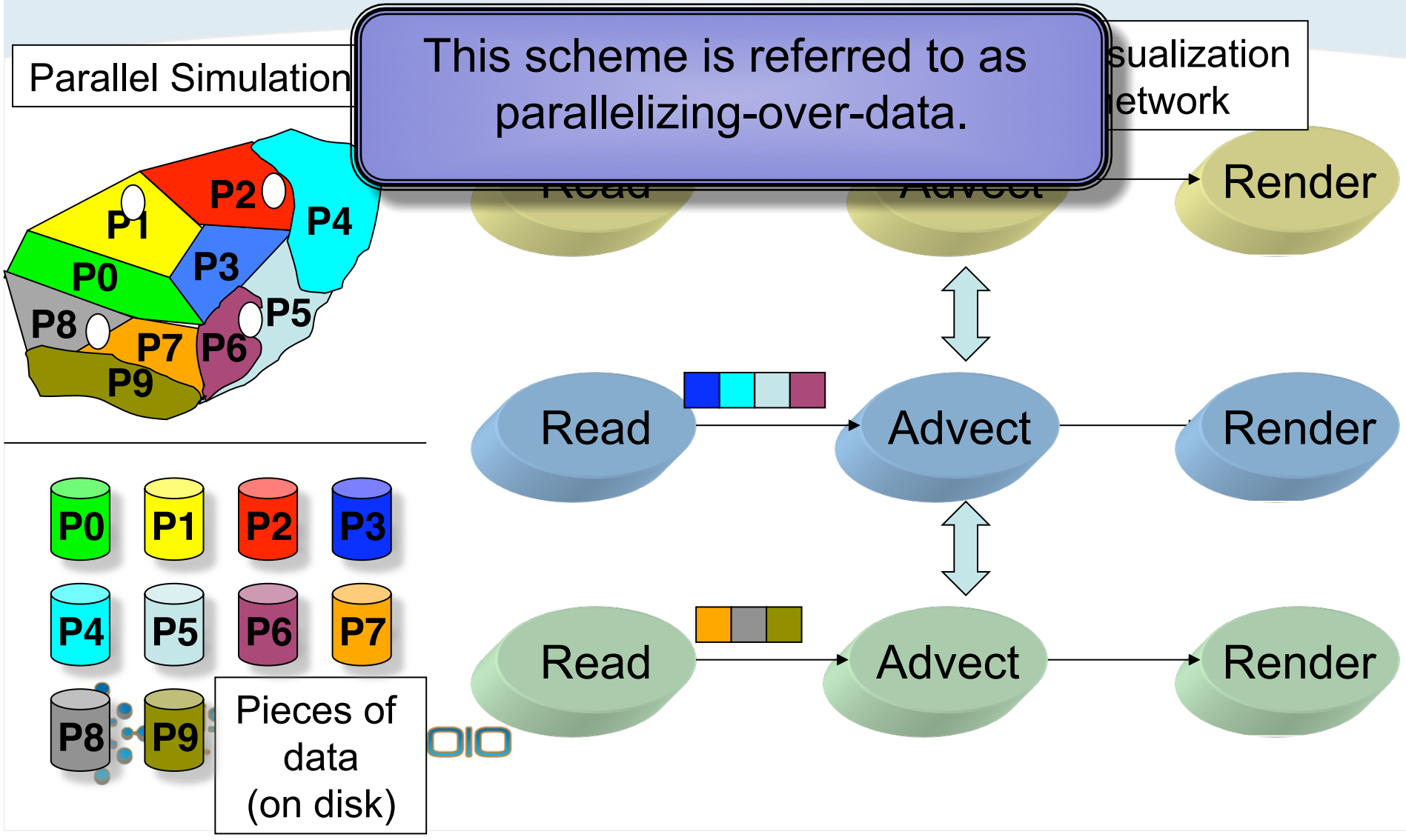


Do we need advanced parallelization techniques? When?

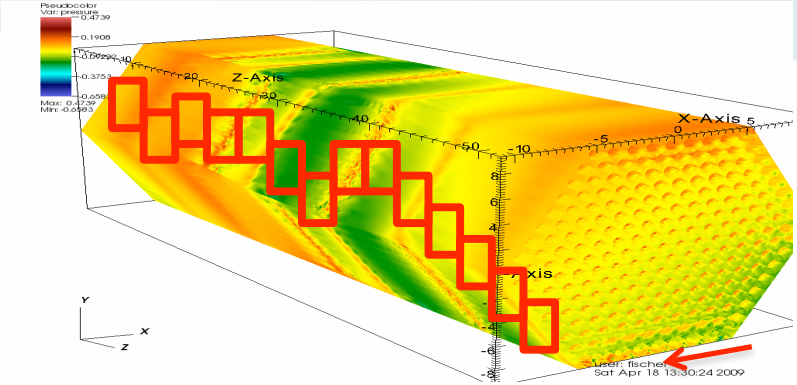


- Data set size?
  - Not enough!
- Large #'s of particles?
  - Need to parallelize, but embarrassingly parallel OK
- Large #'s of particles + large data set sizes

Parallelization for large data with good “distribution”.



Do we need advanced parallelization techniques? When?

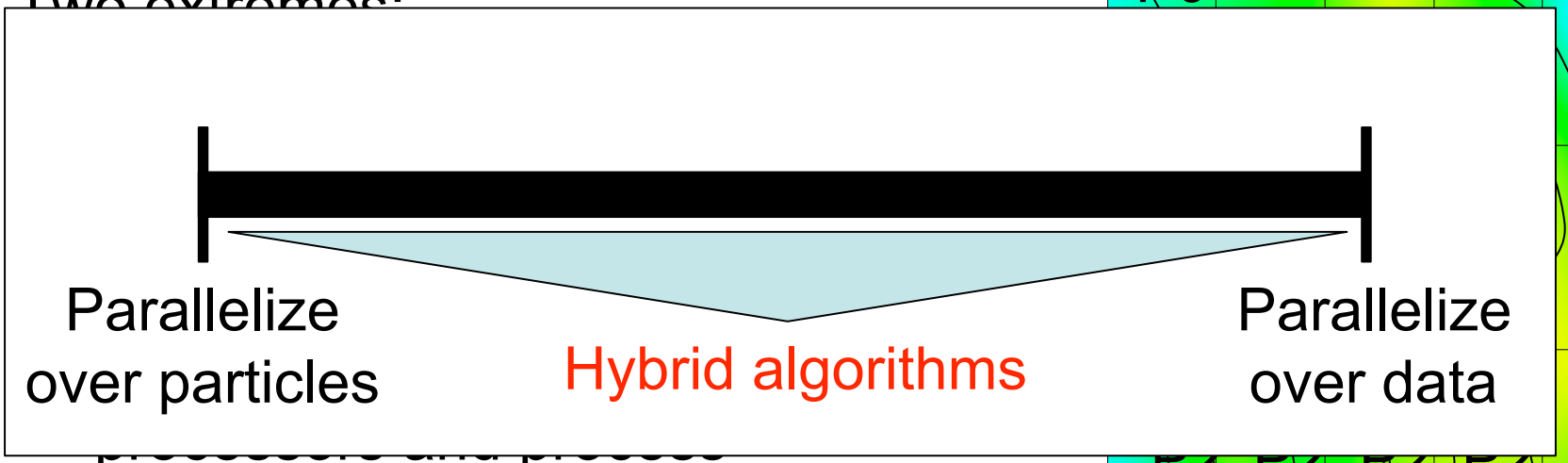


- Data set size?
  - Not enough!
- Large #'s of particles?
  - Need to parallelize, but embarrassingly parallel OK
- Large #'s of particles + large data set sizes
  - Need to parallelize, simple schemes may be OK
- Large #'s of particles + large data set sizes + (bad distribution OR complex vector field)
  - Need smart algorithm for parallelization

VisWeek 2010
VIS • INFOVIS • VAST

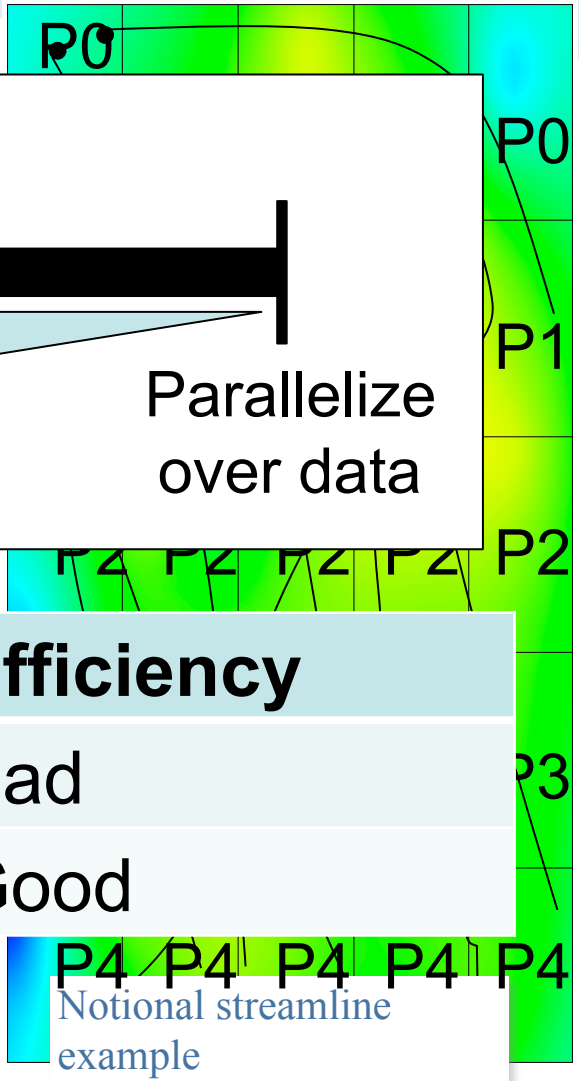
Parallelization with big data & lots of seed points & bad distribution

- Two extremes:



necessary data for advection

Parallelizing Over	I/O	Efficiency
Data	Good	Bad
Particles	Bad	Good



Notional streamline example

Outline

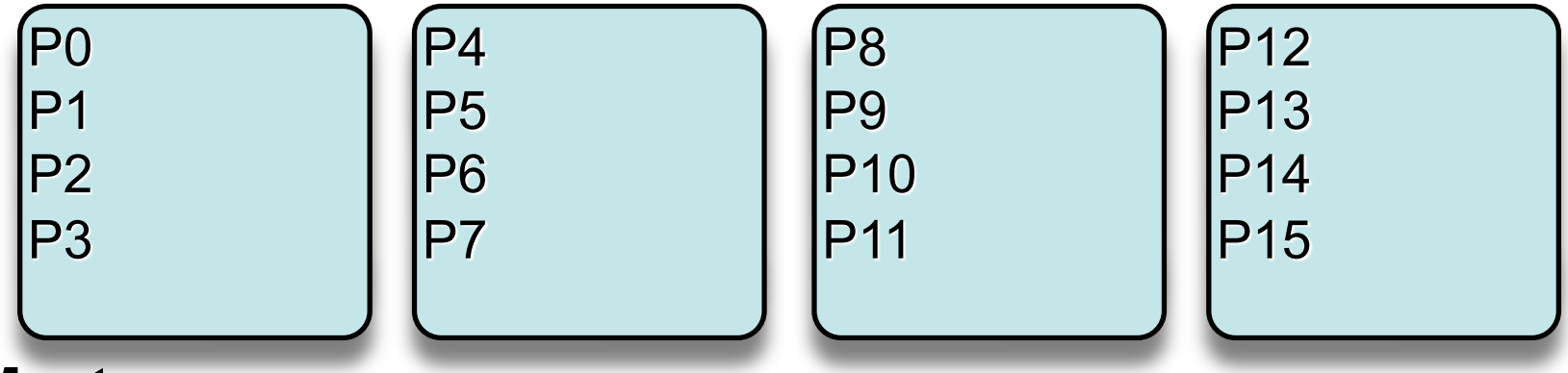
- Motivation
- Parallelization strategies
- Master-slave parallelization
- Hybrid parallelism

The master-slave algorithm is an example of a hybrid technique.

- "Scalable Computation of Streamlines on Very Large Datasets", Pugmire, Childs, Garth, Ahern, Weber. SC09
  - Many of the following slides compliments of Dave Pugmire.
- Algorithm adapts during runtime to avoid pitfalls of parallelize-over-data and parallelize-over-particles.
  - Nice property for production visualization tools.
- Implemented inside VisIt visualization and analysis package.

Master-Slave Hybrid Algorithm

- Divide processors into groups of **N**
- Uniformly distribute seed points to each group



Master:
- Monitor workload
- Make decisions to optimize resource utilization

Slaves:
- Respond to commands from Master
- Report status when work complete

Master Process Pseudocode

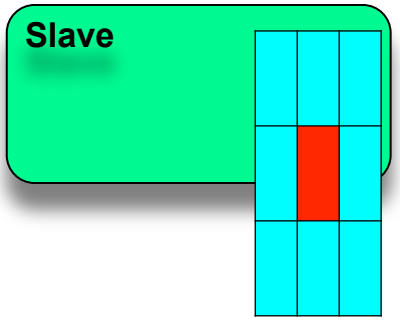
```
Master()
{
    while ( ! done )
    {
        if ( NewStatusFromAnySlave
        {
            commands = DetermineMostEfficientCommand()

            for cmd in commands
                SendCommandToSlaves( cmd )
        }
    }
}
```

What are the possible commands?

Commands that can be issued by master

1. **Assign / Loaded Block**
2. Assign / Unloaded Block
3. Handle OOB / Load
4. Handle OOB / Send

OOB = out of bounds

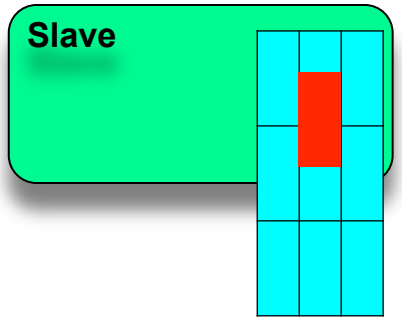


Slave is given a streamline that is contained in a block that is already loaded

Commands that can be issued by master

1. Assign / Loaded Block
2. **Assign / Unloaded Block**
3. Handle OOB / Load
4. Handle OOB / Send

OOB = out of bounds

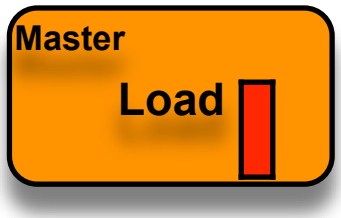


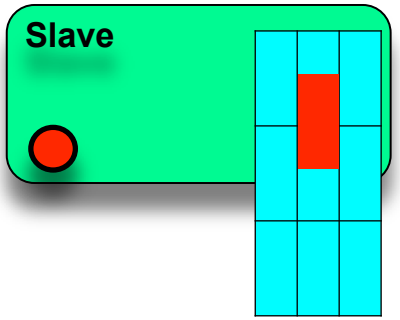
Slave is given a streamline and loads the block

Commands that can be issued by master

1. Assign / Loaded Block
2. Assign / Unloaded Block
3. **Handle OOB / Load**
4. Handle OOB / Send

OOB = out of bounds



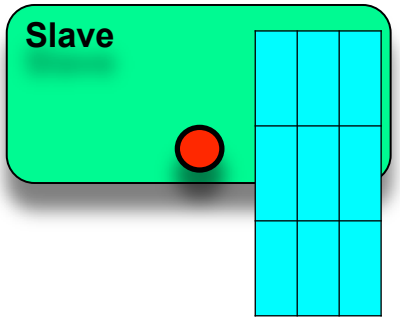


Slave is instructed to load a block. The streamline in that block can then be computed.

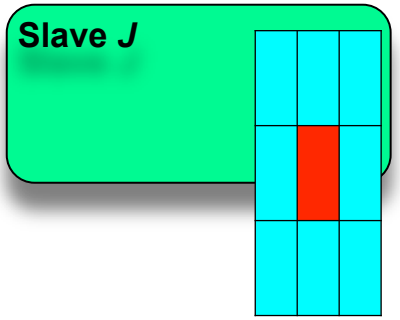
Commands that can be issued by master

1. Assign / Loaded Block
2. Assign / Unloaded Block
3. Handle OOB / Load
4. **Handle OOB / Send**

   OOB = out of bounds



Slave is instructed to send a streamline to another slave that has loaded the block



Master Process Pseudocode

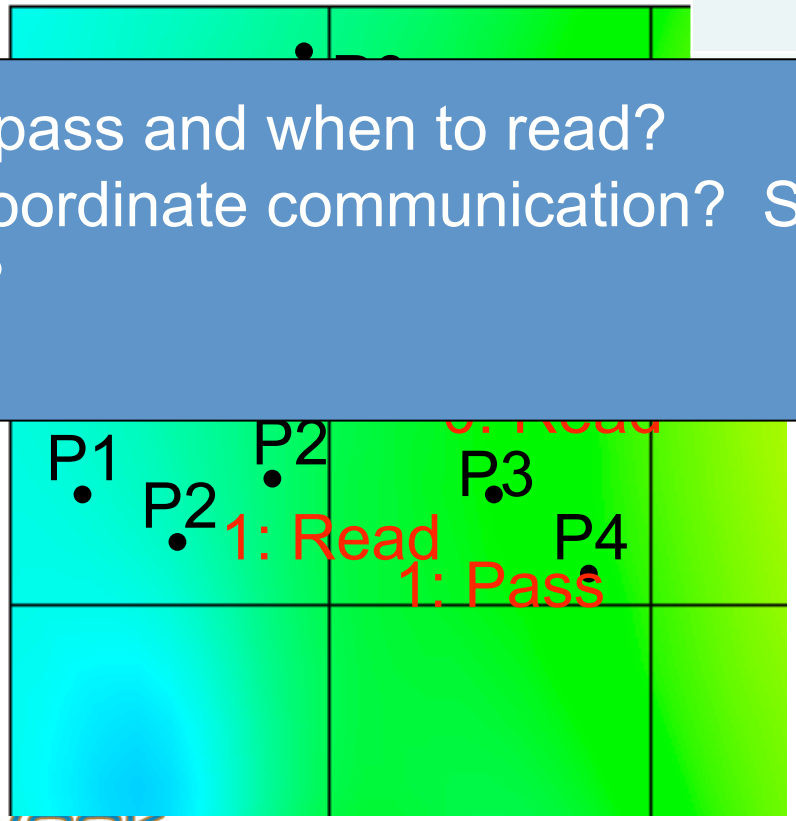
```
Master()
{
    while ( ! done )
    {
        if ( NewStatusFromAnySlave() )
        {
            commands = DetermineMostEfficientCommand()

            for cmd in commands
                SendCommandToSlaves( cmd )
        }
    }
}
```

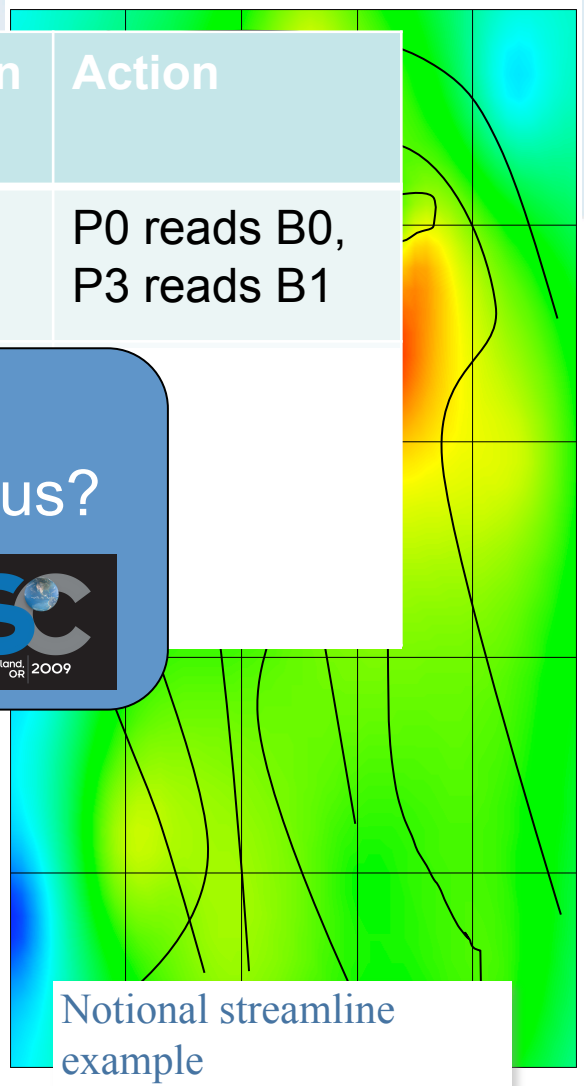
*** See SC 09 paper for details**

Master-slave in action

Iteration	Action
0	P0 reads B0, P3 reads B1

- When to pass and when to read?
- How to coordinate communication? Status? Efficiently?

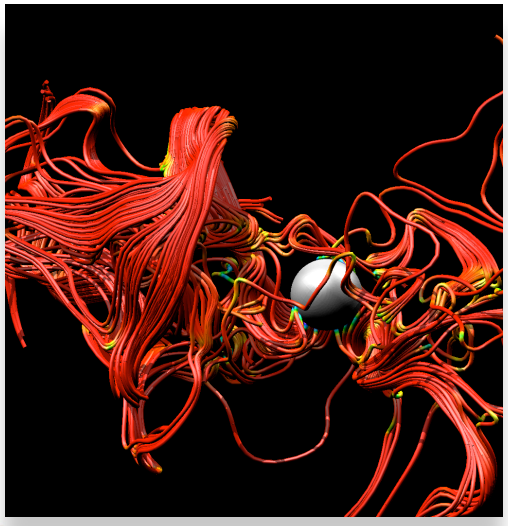


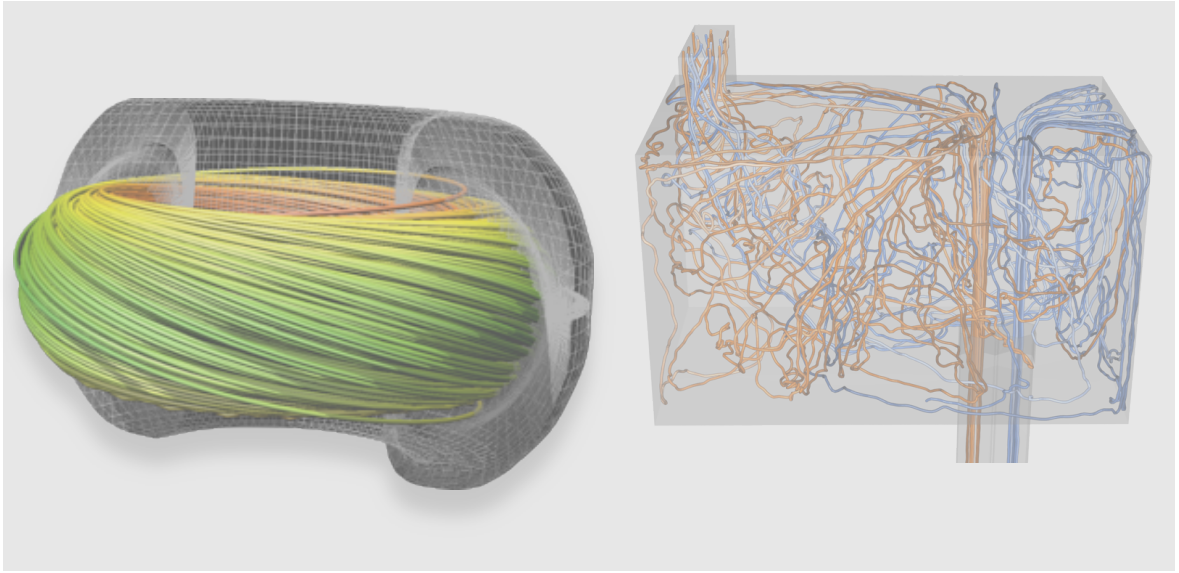


Notional streamline example

Algorithm Test Cases

-Core collapse supernova simulation
-Magnetic confinement fusion simulation
-Hydraulic flow simulation

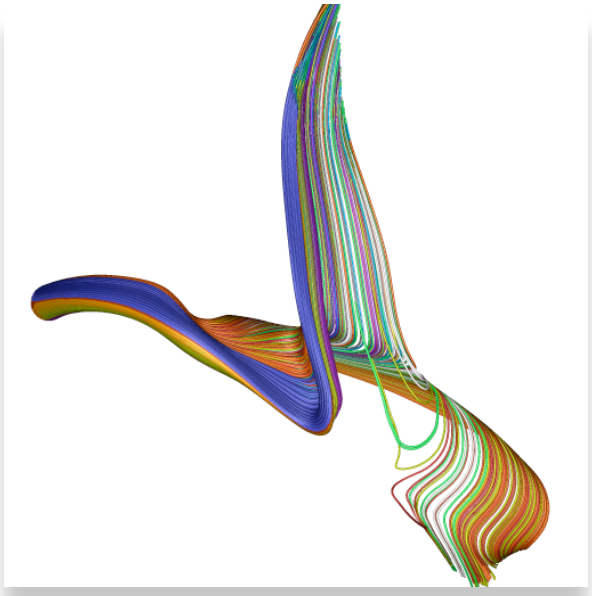




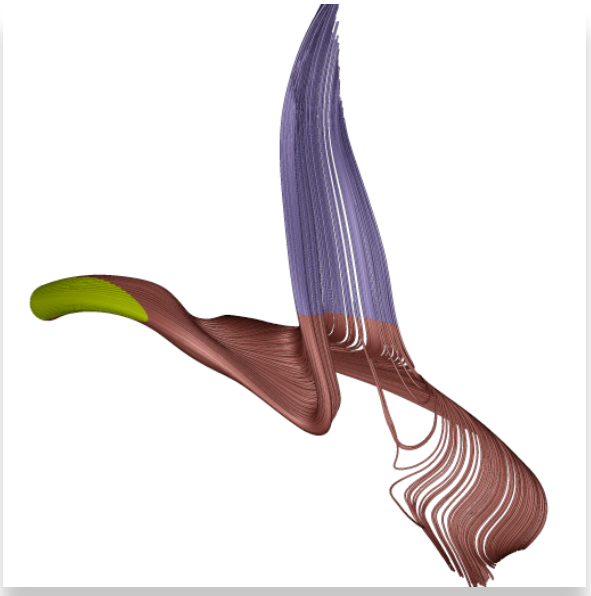
Workload distribution in supernova simulation

Parallelization by:

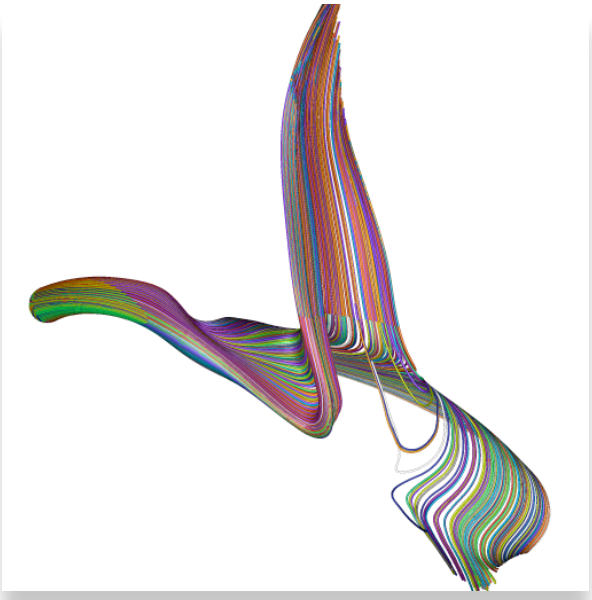
Particles



Data



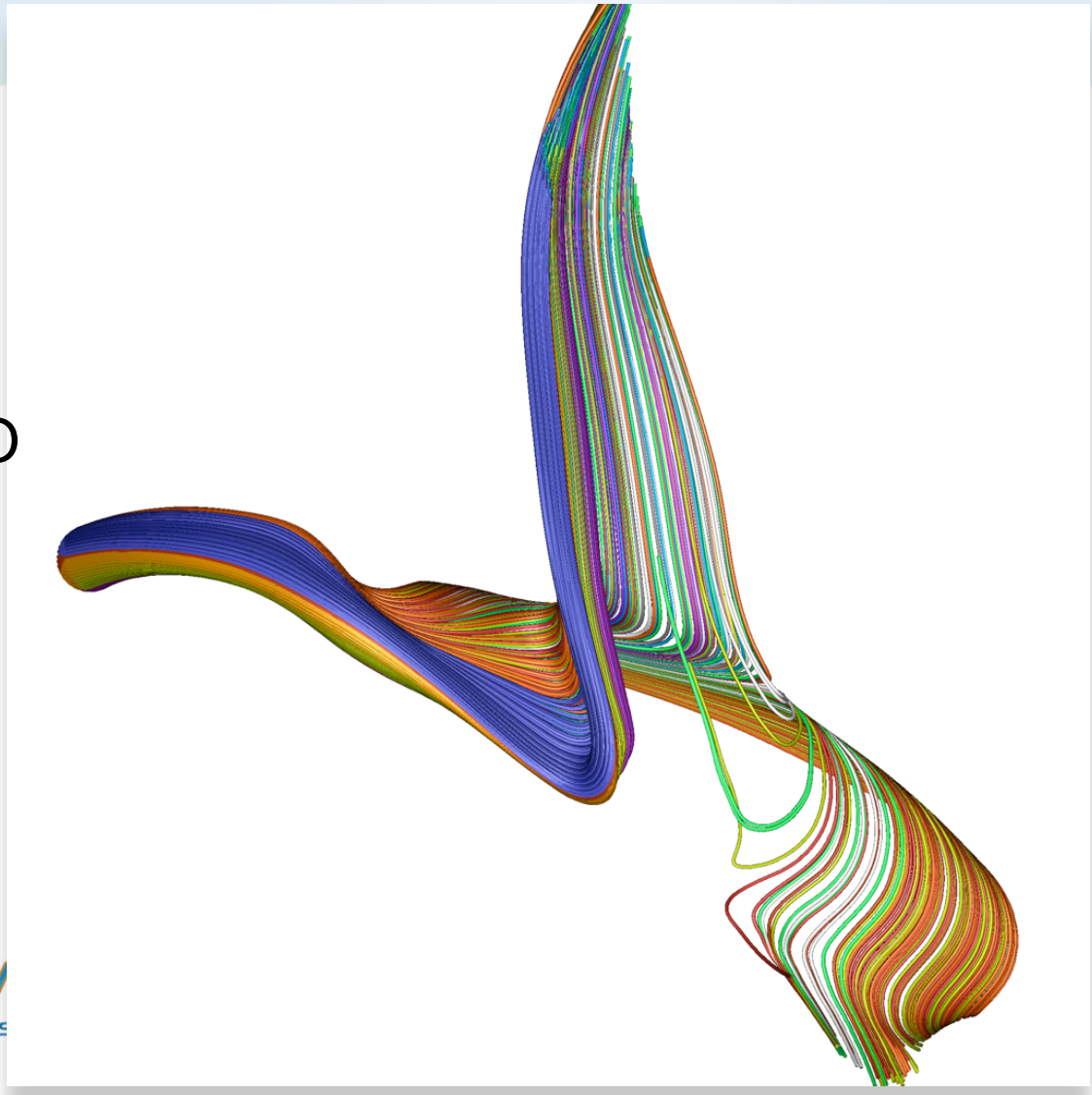
Hybrid



Colored by processor doing integration

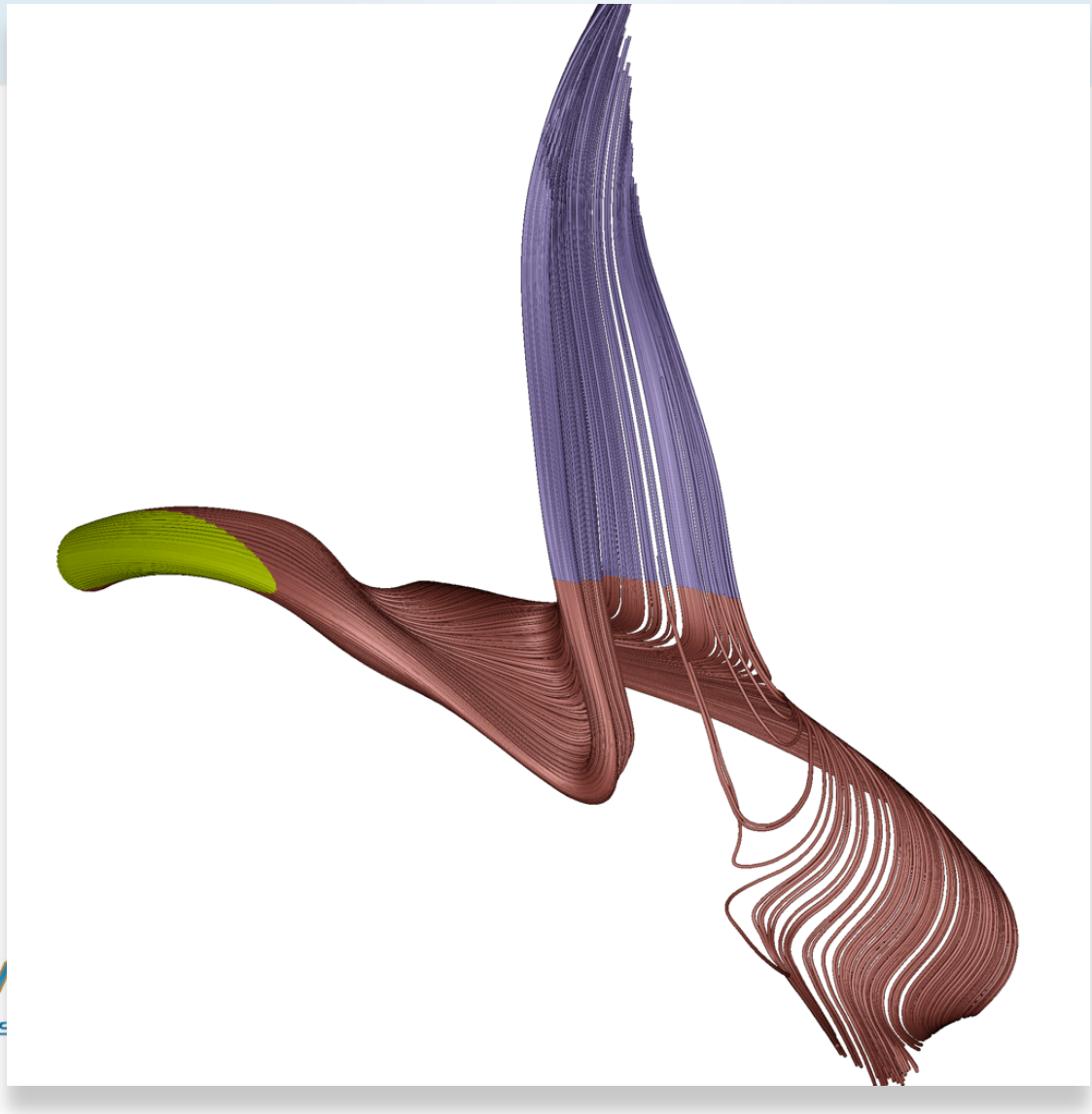
Workload distribution in parallelize-over-particles

Too much I/O



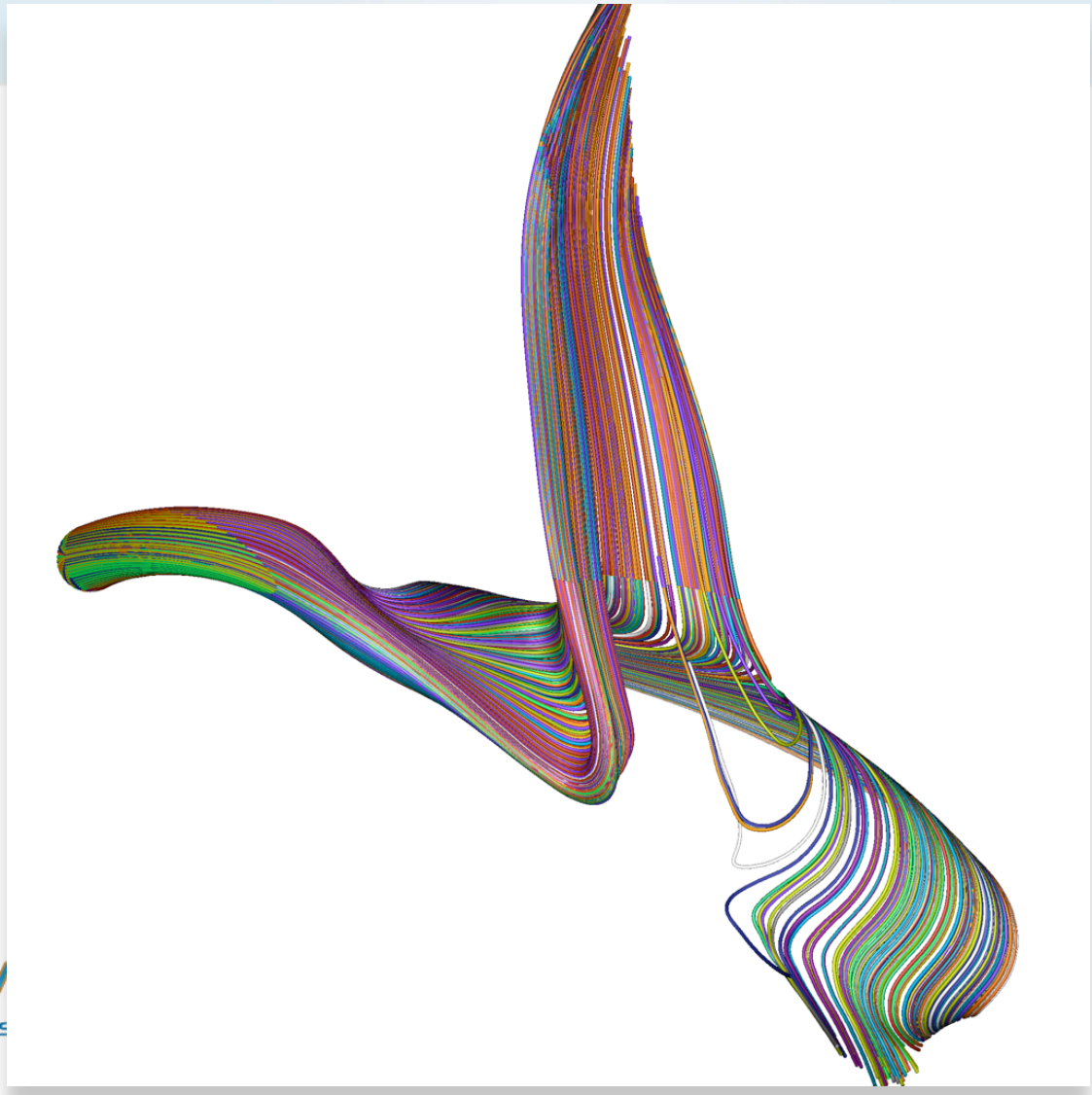
Workload distribution in parallelize-over-data

Starvation

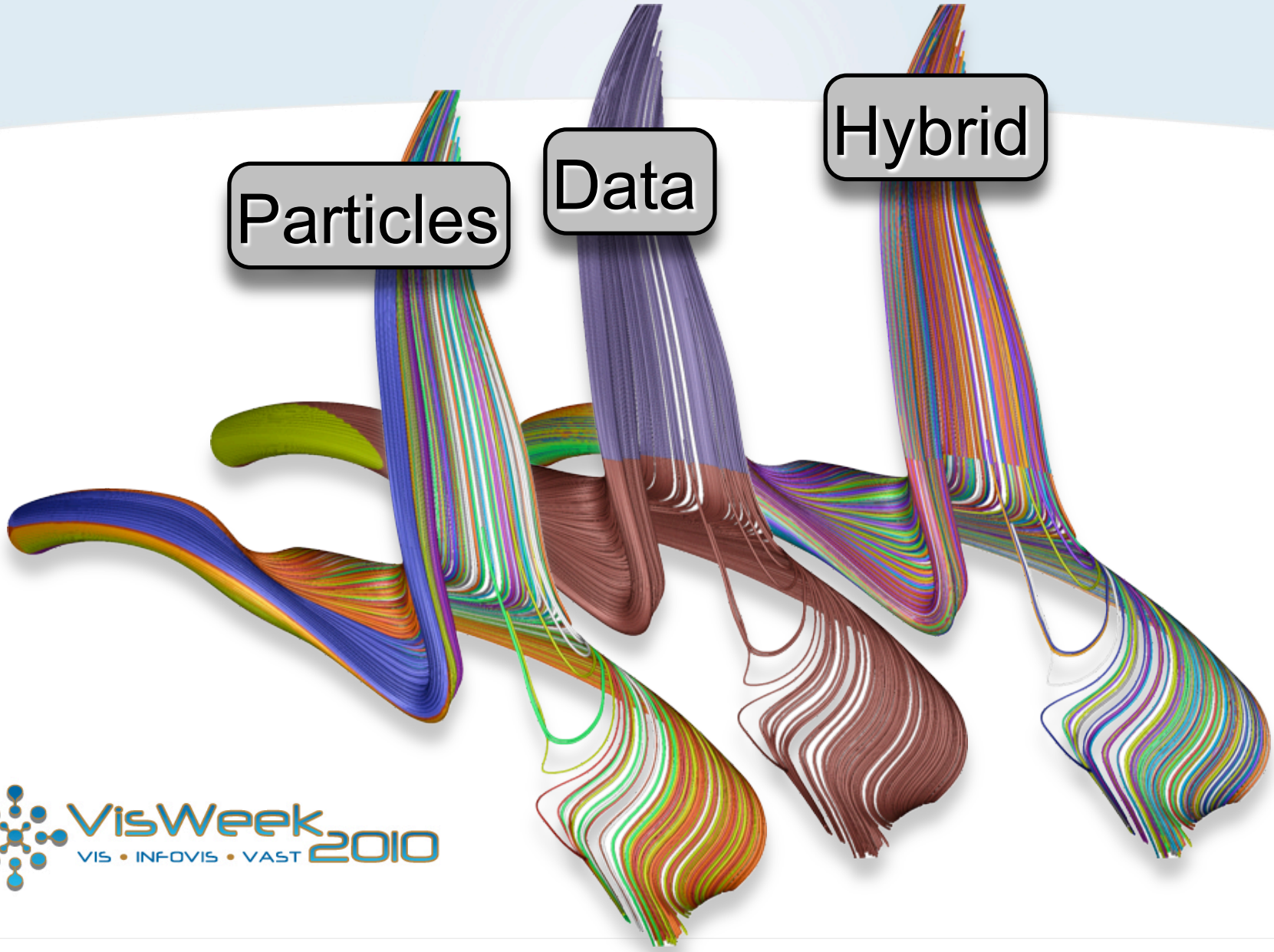


Workload distribution in hybrid algorithm

Just right

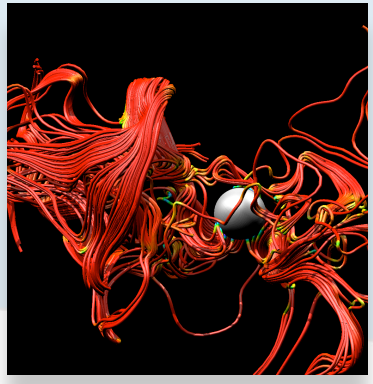


Comparison of workload distribution

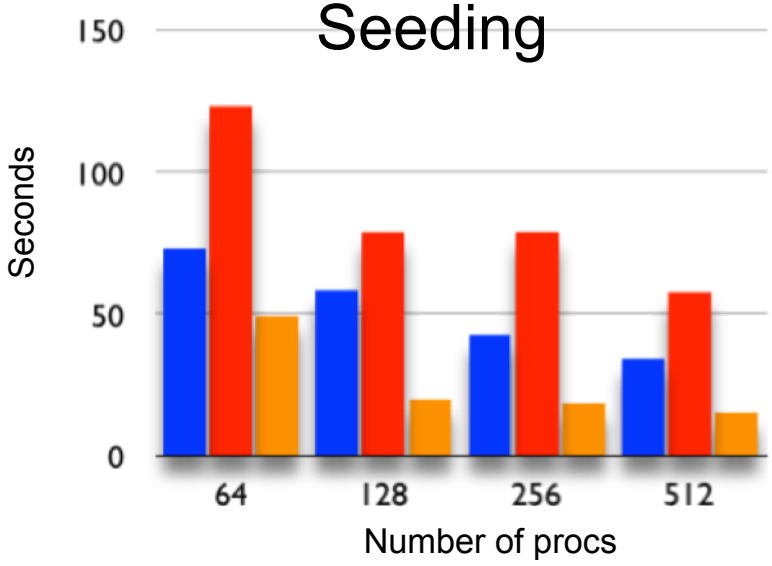


VisWeek 2010
VIS • INFOVIS • VAST

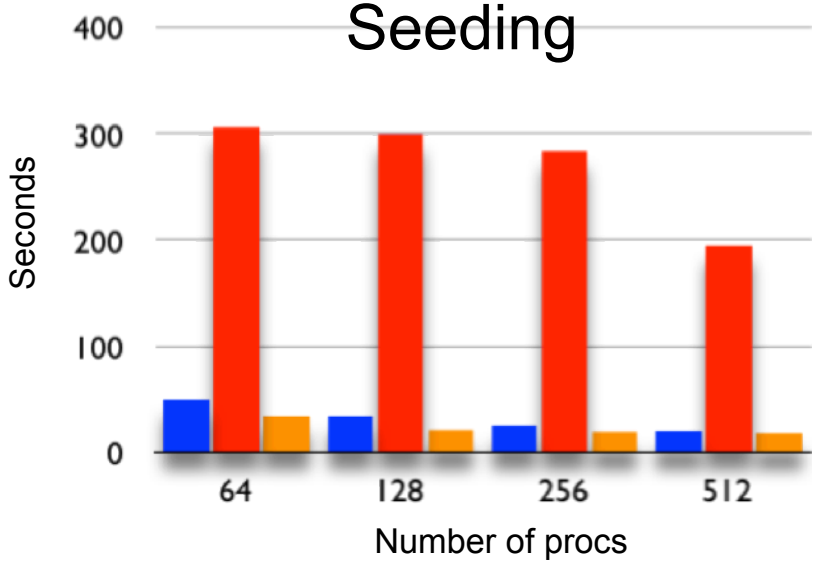
Astrophysics Test Case:
Total time to compute 20,000 Streamlines



Uniform Seeding



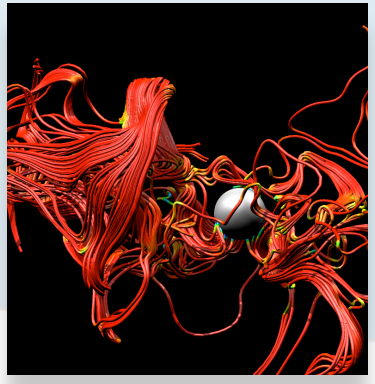
Non-uniform Seeding

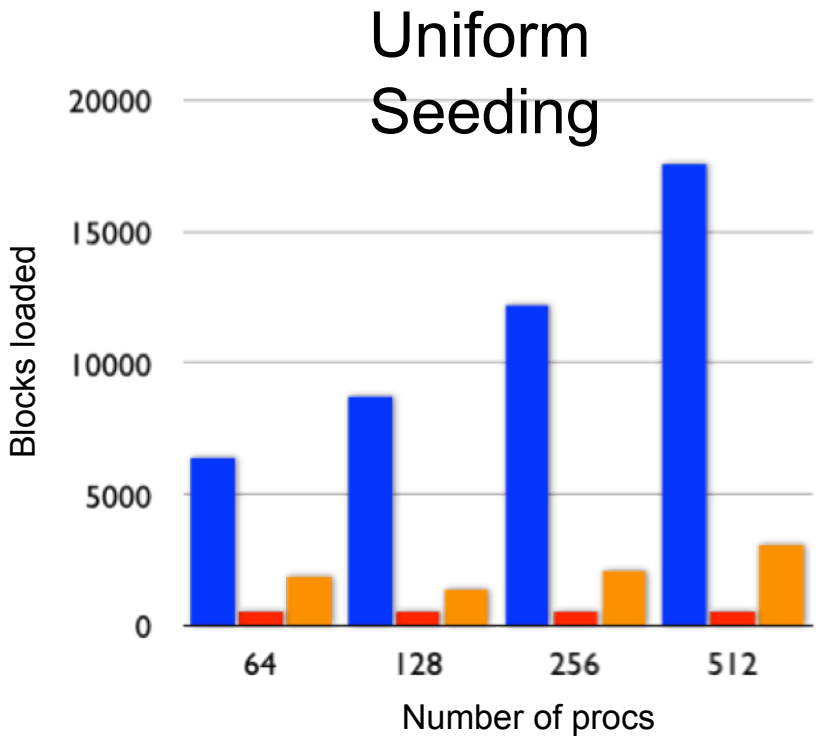


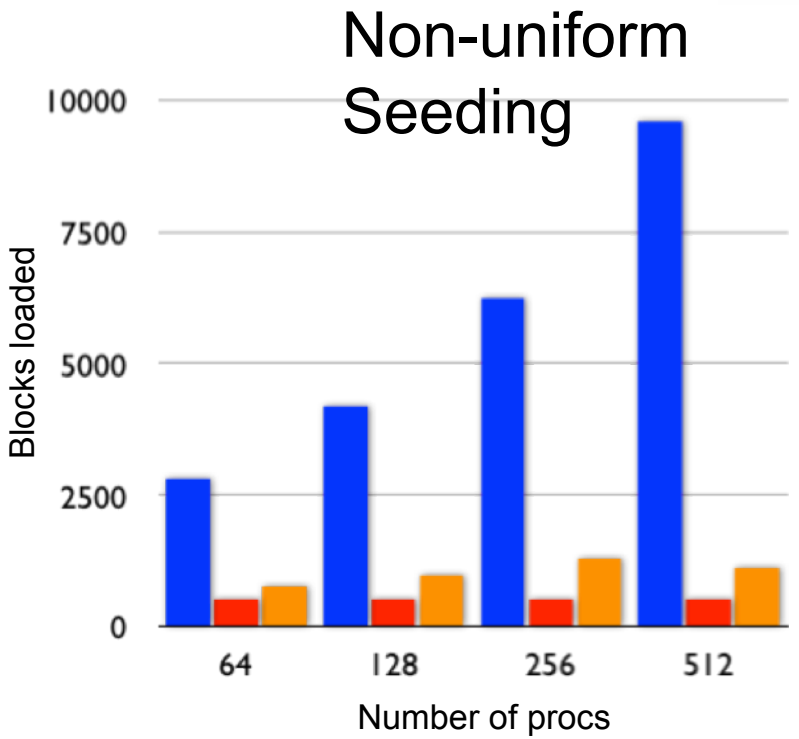
VisWeek 201 VIS • INFOVIS • VAST

Part-icles Data Hybrid

Astrophysics Test Case:
Number of blocks loaded







VisWeek 2010
VIS • INFOVIS • VAST

Part-icles

Data

Hybrid

Outline

- Motivation
- Parallelization strategies
- Master-slave parallelization
- Hybrid parallelism

Are today's algorithms going to fit well on tomorrow's machines?

- Traditional approach for parallel visualization – 1 core per MPI task – may not work well on future supercomputers, which will have 100-1000 cores per node.
  - Exascale machines will likely have O(1M) nodes … and we anticipate in situ particle advection.

Hybrid parallelism blends distributed- and shared-memory parallelism concepts.

The word “hybrid” is being used in two contexts…

- The master-slave algorithm is a ***hybrid algorithm,*** sharing concepts from both parallelization-over-data and parallelization-over-seeds.
- ***Hybrid parallelism*** involves using a mix of shared and distributed memory techniques, e.g. MPI + pthreads or MPI+CUDA.
- One could think about implement a ***hybrid particle advection algorithm*** in a ***hybrid parallel*** setting.

What can we learn from a hybrid parallel study?

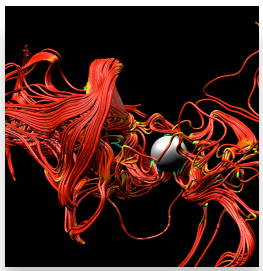
- How do we implement parallel particle advection algorithms in a hybrid parallel setting?
- How do they perform?
  - Which algorithms perform better? How much better?
  - Why?

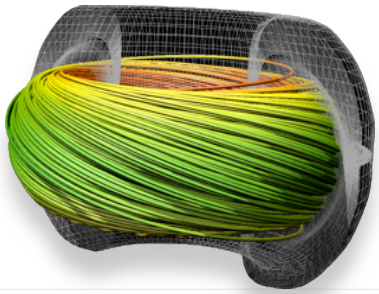
Streamline Integration Using MPI-Hybrid Parallelism on a Large Multi-Core Architecture
by David Camp, Christoph Garth, Hank Childs, Dave Pugmire and Ken Joy
Accepted to TVCG

Streamline integration using MPI-hybrid parallelism on a large multi-core architecture

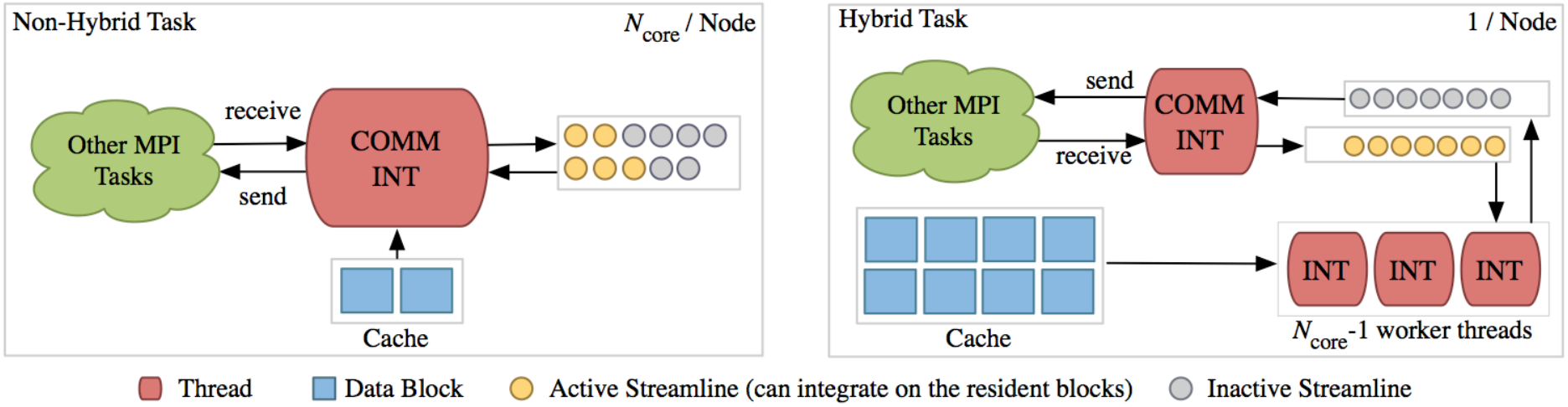
- Implement parallelize-over-data and parallelize-over-particles in a hybrid parallel setting (MPI + pthreads)
  - Did not study the master-slave algorithm
- Run series of tests on NERSC Franklin machine (Cray)
- Compare 128 MPI tasks (non-hybrid)
  vs 32 MPI tasks / 4 cores per task (hybrid)
- 12 test cases: large vs small # of seeds
  uniform vs non-uniform seed locations
  3 data sets

VisWeek 2010
VIS • INFOVIS • VAST

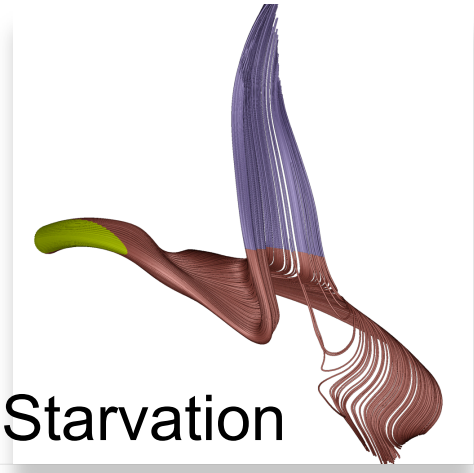




Hybrid parallelism for parallelize-over-data

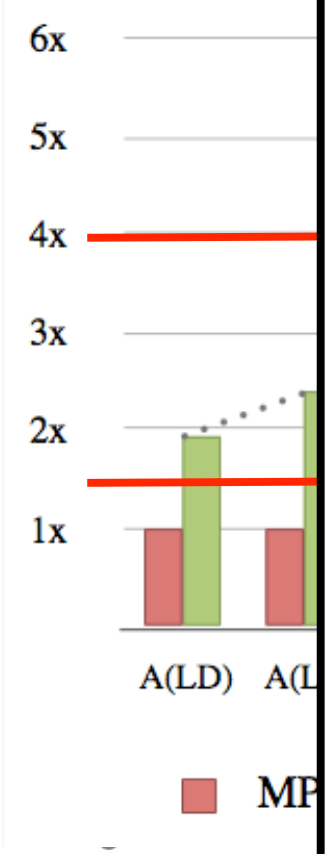


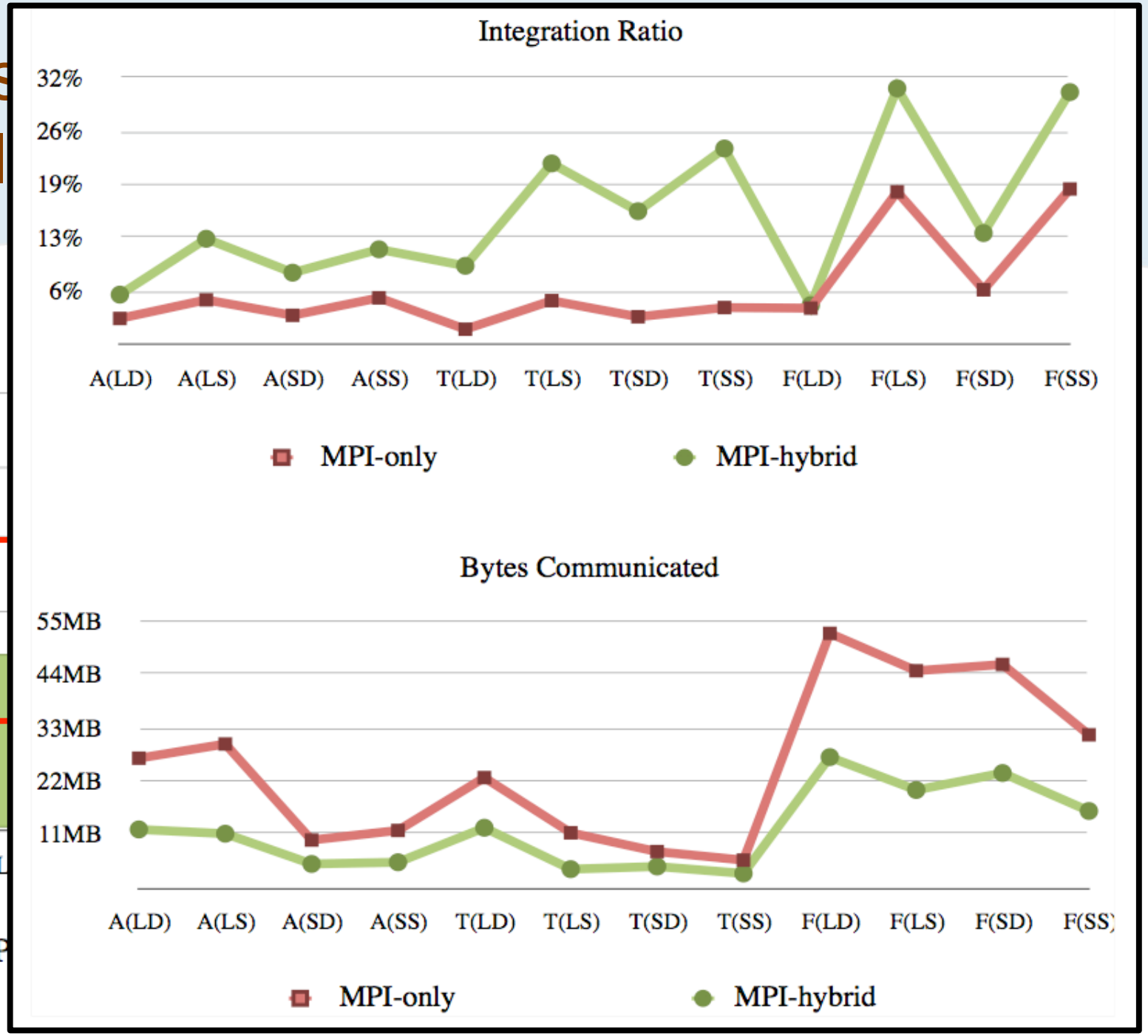
- Expected benefits:
  - Less communication and communicators
  - Should be able to avoid starvation by sharing data within a group.



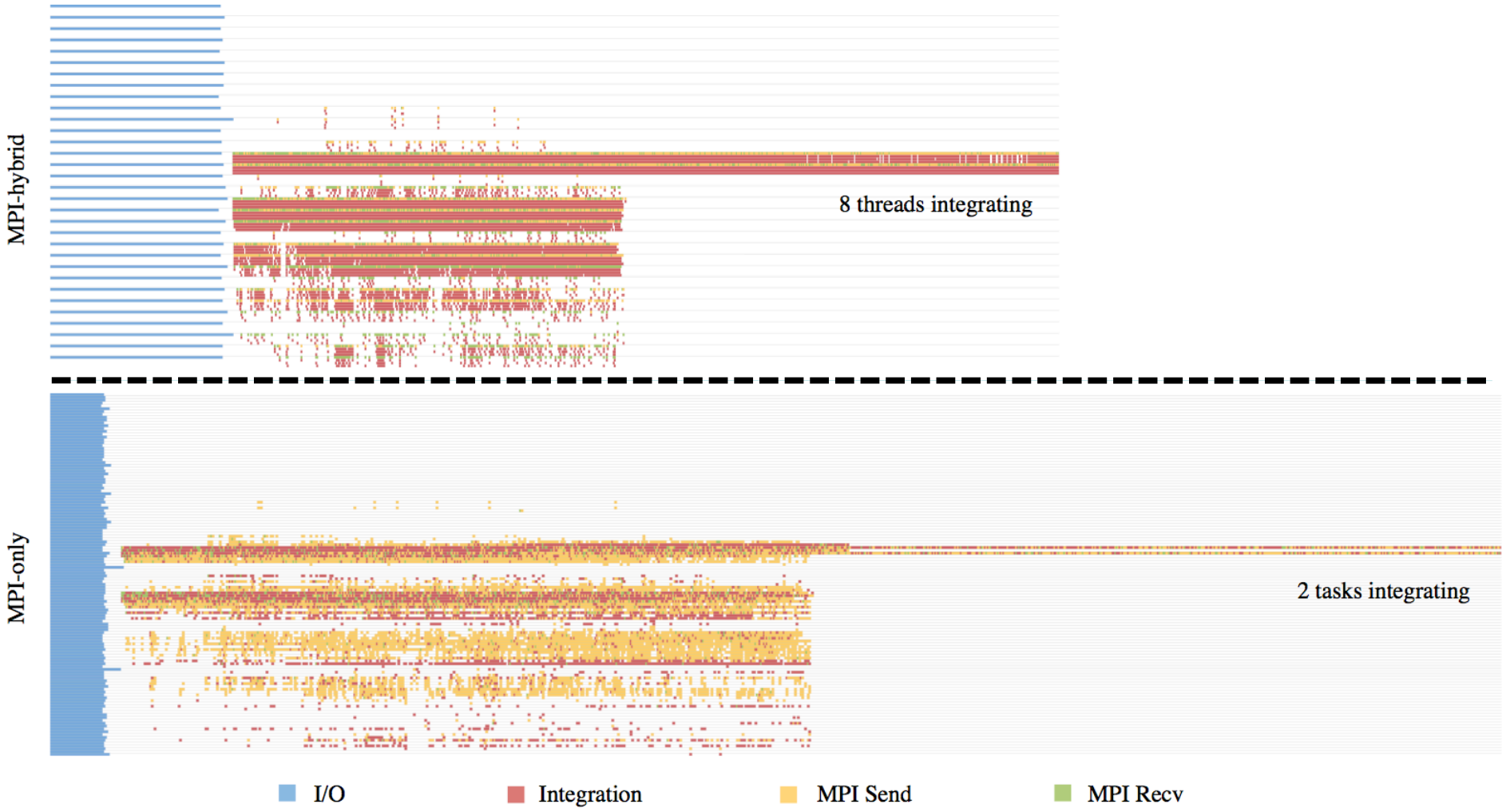
VisWeek 2010
VIS • INFOVIS • VAST

Meas
paral


6x
5x
4x
3x
2x
1x
A(LD)
A(L
MP

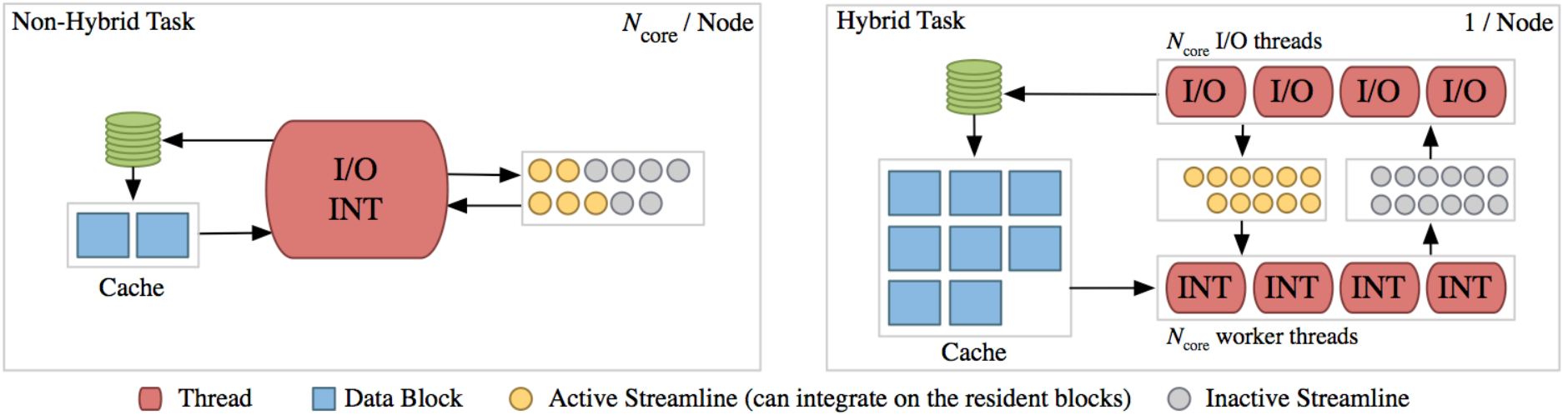

Integration Ratio
32%
26%
19%
13%
6%
A(LD) A(LS) A(SD) A(SS) T(LD) T(LS) T(SD) T(SS) F(LD) F(LS) F(SD) F(SS)
MPI-only
MPI-hybrid
Bytes Communicated
55MB
44MB
33MB
22MB
11MB
A(LD) A(LS) A(SD) A(SS) T(LD) T(LS) T(SD) T(SS) F(LD) F(LS) F(SD) F(SS)
MPI-only
MPI-hybrid

Gantt chart for parallelize-over-data



VisWeek 2010
VIS • INFOVIS • VAST

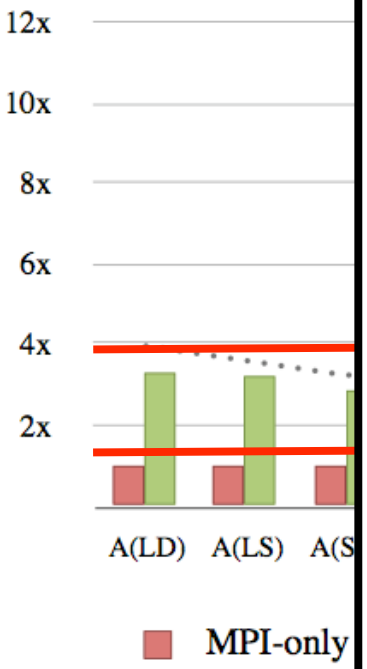
Hybrid parallelism for parallelize-over-particles

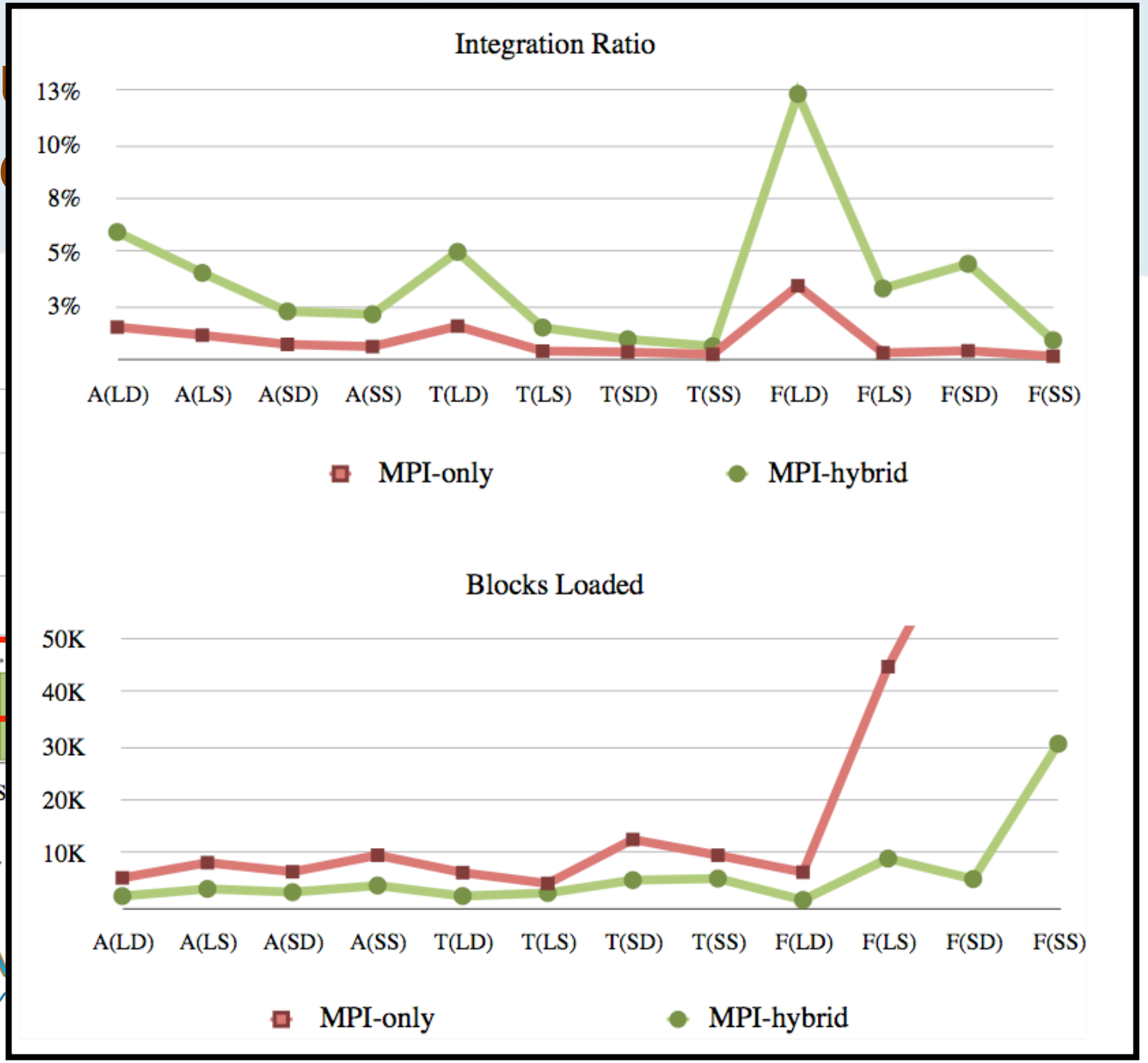


- Expected benefits:
  - Only need to read blocks once for node, instead of once for core.
  - Larger cache allows for reduced reads
  - “Long” paths automatically shared among cores on node

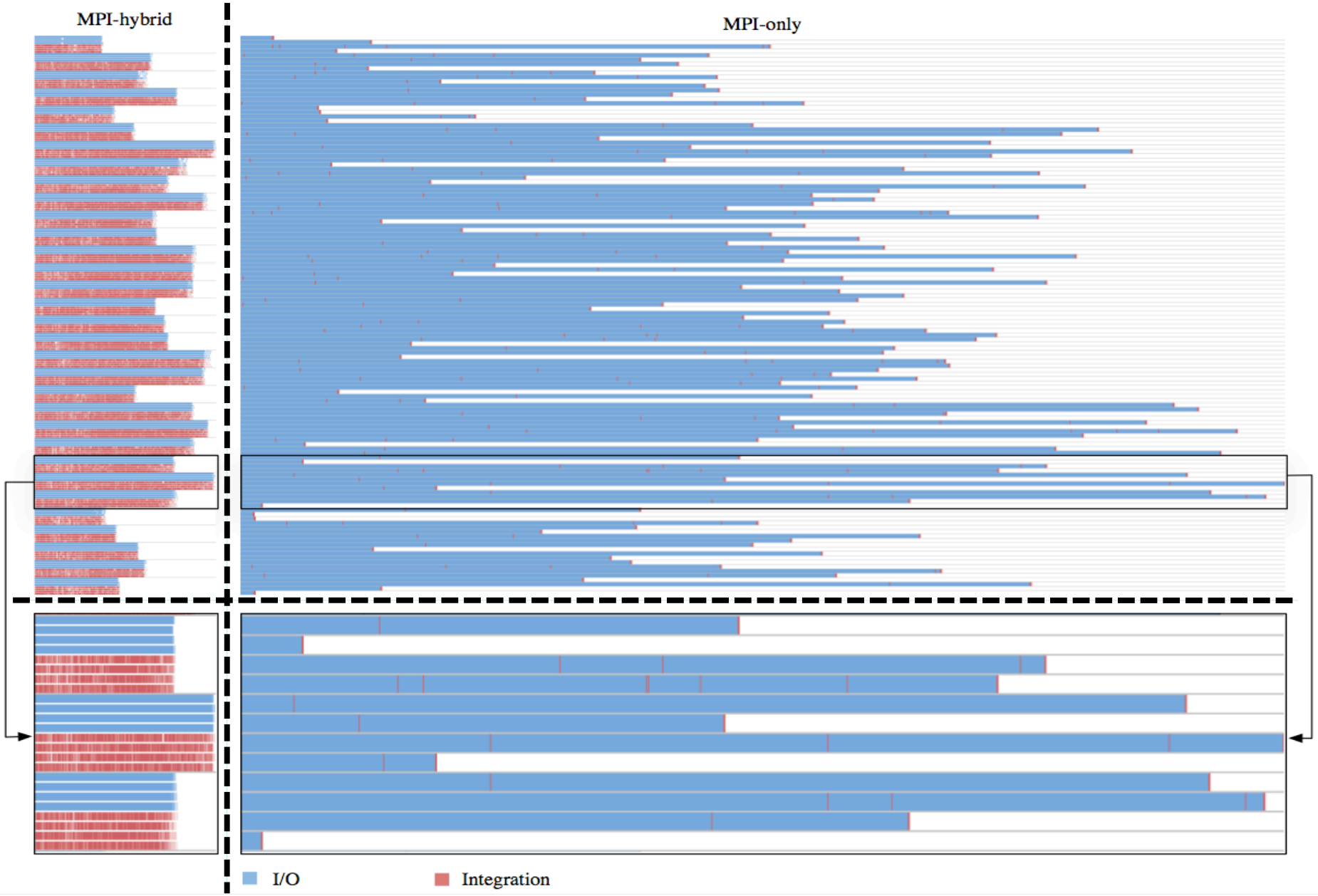
VisWeek 2010
VIS • INFOVIS • VAST

Meas
paralle





Gantt chart for parallelize-over-particles



Summary of Hybrid Parallelism Study

- Hybrid parallelism appears to be extremely beneficial to particle advection.
- We believe the parallelize-over-data results are highly relevant to the in situ use case.
- Although we didn't implement the master-slave algorithm, we believe the benefits shown at the spectrum extremes provide good evidence that hybrid algorithms will also benefit.
- Implemented on VisIt branch, goal to integrate into VisIt proper in the coming months.

Summary for Large Data and Parallelization

- The type of parallelization required will vary based on data set size, number of seeds, seed locations, and vector field complexity
- Parallelization may occur via parallelization-over-data, parallelization-over-particles, or somewhere in between (master-slave). Hybrid algorithms have the opportunity to de-emphasize the pitfalls of the traditional techniques.
- Hybrid parallelism appears to be very beneficial.
- Note that I said nothing about time-varying data…